0990706.051101

LOST2 - A POSITIONING SYSTEM FOR UNDERWATER VESSELS

0990570 6.051101

09905706.051101

09905706.051101

0990706 .051101

Chapter 1

Introduction

Navigating the open ocean was one of the hardest scientific problems in the 18th and 19th centuries[1]. Without the ability to determine their position accurately, sailors were lost from the time the sight of land was lost until the next landmass was spotted. Navigators, unaware of their actual position, would often run aground on either known or unknown rock shoals. Then, as now, most of the world's commerce travels at least part of the way on the sea, making navigation a serious problem. Today, with the global positioning system, determining the position of most objects is a simple task. However this system is not available underwater or on other worlds, leaving the only options available to the sailors of old, primarily integrating velocity and heading measurements. This dissertation is devoted to the development of a new method to position underwater vessels accurately. The navigation problem is reduced to the problem of accurately determining the position of the body at any given time.

0990706.051101

1.1 The Navigation Problem

Leonard and Durrant-Whyte [2] sum up the problem of navigation by answering three questions: "where am I?", "where am I going?" and "how do I get there?". In order to answer these questions a coordinate system is required. It does no good to say go to point $(254, 254)$ from point B if they are referenced to two different coordinate systems. A coordinate system that relates the environmental conditions to the coordinates is referred to as the map of the environment. Therefore, the first three questions can be answered only in terms of a forth "what is my map?" Because the first and forth questions define the answers to the second and third, navigation is also referred to as the mapping and localization problem. Most of this dissertation is devoted to solving the localization part of the navigation problem.

The question "where am I going?" should be a mission parameter that is defined by the mission goal. This goal is usually determined before deployment and therefore is not addressed in this document.

Given a map and a means to perform the localization of an object, a path from where an object is to where it wants to go can be plotted. This path needs to be collision free, i.e., this path is an answer to the question, "how do I get there?" This problem is referred to as path-planning and has been well addressed in the robotics literature, and so is not discussed in this document.

The task of making accurate maps of the environment is something that can be easily accomplished using today's technologies. These technologies are generating maps of higher and higher resolution. The main outstanding issue is knowing exactly where the sensor that made the measurements for the map was in order to correctly render the map. Because the mapping problem requires the solution to the localization problem, the fundamental problem in the area

0990706.051101

of navigation is then "where am I?" or the localization problem. This problem has been mostly solved for air, land and surface based systems with the use of the global positioning system. This leaves underwater applications as the major terrestrial area where the localization problem has yet to be solved. There are three general categories of solutions to this problem: dead - reckoning, acoustic based positioning and terrain based localization. The next section describes these methods in detail.

1.2 Importance of Localization for Underwater Vessels

The localization problem is particularly challenging for underwater vessels because electro-magnetic signals can penetrate only a few meters except at very low frequencies, limiting their usefulness. Sound propagates well in the underwater environment enabling long range sensing and positioning systems. Sonar signals however suffer from a low bandwidth and are subject to high levels of noise. Underwater vessels are also subject to unknown forces that are neither predictable nor readily able to be modeled. However, there is a great potential for underwater vessels in the scientific, commercial and military realms if the positioning problem can be solved.

The use of underwater vehicles for scientific purposes has been advocated for some time [3, 4, 5]. For example MIT has developed an AUV for scientific missions called the Odyssey IIB. This AUV is shown in figure 1.1. Unmanned underwater vessels have the advantage of being able to place sensors closer to the objects of interest on the sea floor much cheaper than using a submarine or other manned vehicle. Unmanned vessels can also enter situations that are

too dangerous for a manned scientific mission. The potential for unmanned underwater vehicles is remarkable, especially considering the reduction in cost over present methods.

Commercially, there are many situations in which accurate positioning of underwater vessels would allow for the mission to be performed more efficiently. For example on many cable runs in the open ocean, companies will tow a side scan sonar to produce an image of the ocean floor to locate a safe path for the cable. Due to the lack of an self contained accurate positioning system, a second vessel is often required to remain over the tow in order to provide for accurate positioning. A large cost savings can be realized if the need for the second vessel is removed. As another example of the positioning problem, consider the Hugin unmanned underwater vehicle which performs bathymetric mapping at depths up to 600 meters[6]. This vessel requires a surface ship to remain inside a 30 degree cone in order to provide the positioning via an acoustic modem. Any reduction in the hardware required for the positioning problem would be of great benefit to commercial operations.

The potential for military applications of unmanned underwater vessels is limited only by the creativity of the users. The most obvious are in covert operations, surveillance, and mine-countermeasures. The goal of mine-counter measures is to detect, localize and neutralize mines [7]. Traditionally, mine counter measures has required the use of manned vessels to go in and sweep for mines. After the mines are located, then a group of divers is required to neutralize the mines. If there were an accurate navigation system this whole process could be done autonomously, reducing the number of people exposed to hazardous conditions. Unmanned vessels also have the potential to be used in covert operations.

0990706.051101

1.3 Current Methods of Underwater Positioning

Depending on the application, the position of an underwater body can be determined by dead-reckoning or an acoustic based positioning system. Recently, the concept of using terrain-based navigation has gained popularity[8, 9, 10, 11, 12, 13].

1.3.1 Dead-Reckoning

The most traditional method of navigating underwater some form of dead-reckoning. This falls into two primary categories, velocity based positioning and system models. Velocity based positioning is used primarily for AUV's, while system models are used for towbodies.

The most obvious navigation method is dead-reckoning, where the vehicle's velocity is integrated over time to estimate location[14]. Measurement of velocity is usually done by means of a compass and a log. The problem with this is twofold. First, the presence of currents are not accounted for by the log. This can lead to errors in position as large as five nautical miles per hour of operation. The second problem is that the errors in measurement of the velocity also get integrated with respect to time causing the system to drift. When the vessel is near the seafloor, a Doppler velocity log can be used to measure the velocity of the vessel with respect to the seafloor. This can be coupled with a Kalman filter to smooth the velocity estimate to improve the position estimate.

As an alternative to measuring velocity, inertial navigation systems exists In which the accelerations of the body are measured and then integrated twice with respect to time to generate position[15]. Cost and power have traditionally prevented these systems from being used on smaller underwater vessels, but this might change if inertial systems get smaller and cheaper.

0990506.051101

The problem with exclusive reliance on these method is that the error increases without bound as the length of the mission increases. The rate depends on the exact system setup, but after a long enough period of time, the position is useless.

An alternative method shown in Figure 1.2is used in many towbody operations. This methods uses models of the cable system to determine the position of the vessel. At its simplest, the cable layback model method is based on an assumption that the towbody remains behind the towing vessel at all times and the cable is a straight line from the vessel to the towbody. Locating the towbody is a matter of simple geometry. The assumptions of this model are rather poor in practice; currents in the water, ship movements such as turning or changing speed and cable drag and weight cause catenaries and offset in the cable. More complex models for cable layback are possible[16]; however, Syck[17] performed a detailed analysis of several different cable models and concluded that more complex models do not provide better accuracy. In general, these methods provide estimates of location with errors on the order of 10% of cable length for long(3-4 km) cable runs. Because of the large error associated with the cable layback models, it is common to use a simple rule such as 3 meters back for every meter deep. This method is.

Using these models has the advantage of not requiring any external hardware or power consumption on the vessel, but the positioning is poor. For missions which require accurate positioning, cable models and velocity based methods are not accurate enough and other methods must be used.

0990706.051101

1.3.2 Acoustic Methods

There are two acoustic methods used to locate underwater objects in current use, long baseline (LBL) and short baseline (SBL). These are described in detail by Blondel and Murton[18]. Both rely on transponders and estimate position by acoustic ranging, and the difference between the two methods is that LBL performs triangulation while SBL uses a single range and angle to perform the positioning.

The first acoustic method, LBL, is shown in Figure 1.3. Before any positions can be provided by an LBL system, transponders must be deployed and precisely located[19, 20]. This can be done by placing each transponder and then taking a range from multiple known points to triangulate the transponder's location. This requires a large amount of time and effort to set up the transponders before surveying operations can begin. LBL transponders can also be mounted on GPS equipped buoys, saving time on setup[21]. The location of the vessel is then determined by triangulation using the range from the various transponders. The vessel must remain within acoustic range of the transponders at all times to ensure accurate navigation. The setup of the transponders is expensive, and when the operational area is changed the transponders must be moved.

The other method, SBL, is shown in Figure 1.4. SBL uses hydrophones on either the object or on another vessel and a transponder on the other. There are two forms of SBL in use today. These are the 'normal' SBL where the hydrophone net is placed on the surface vessel and the inverted SBL where the hydrophone net is placed on the underwater vessel. These two systems work in the same basic manner. When the signal from the transponder reaches the net the time delays are converted to distance and the difference in the arrival times at each element is used to calculate the angle. Typically, this allows the vessel to be located to

0990706.051101

within 25 meters at ranges of up to 2.5 kilometers[22]. The main source of error in this setup is due to angular inaccuracies in measurements, often one degree or more. For very deep operations, this method requires a surface vessel to remain over the vessel in order to maintain sufficient accuracy. Even this modification is subject to angular errors, but the effect is reduced due to the reduction in range from the transponder to the vessel. The amount of hardware and the occasional need for multiple vessels adds to the costs of using the system.

1.3.3 Terrain Based Navigation

For underwater operations it is desirable to not be reliant on surface ships or large scale transponder nets prepositioned on the ocean floor. If an accurate a priori map of the operational area is available, a new approach to globally-referenced positioning is to use measurements of geophysical parameters such as bathymetry, magnetic field or gravity fields[23, 24, 25, 26]. These approaches attempt to match the current sensor readings with the a priori map of the area, assuming that there is sufficient spatial variation in the parameter being measured to permit the localization. The current methods of terrain matching fall into three general categories: grid-based, feature-based and topological.

Grid Based Navigation

The grid based approach to the terrain matching problem is the easiest to conceptualize[27]. This approach is referred to as the metric approach [28, 29]. In these approaches, the environment is divided into a grid of cells of some fixed size. Each cell in the three dimensional grid is then assigned a value between 0 and 1 depending on how much of it is occupied. This algorithm will produce a map that gives each grid cell a number based on its occupancy measurement function. Maps of this form

0990706.051101

are called certainty grids. To locate a vessel, the sensors generate a certainty grid for the vessel's surrounding. A search is then performed to find the best match (correlation) between the observed and the stored grid spaces. The match with the highest correlation is assumed to be the location of the vessel.

This method has several advantages as well as disadvantages. The major advantages of grid based methods are their inherent simplicity, they are easy to implement and they extend to higher dimensions. However, while these methods are useful in certain environments, differentiating between similar environments is difficult. These methods also experience high computational costs and require large amounts of storage.

This method is being developed for use on other planets as well as on the Earth. Clark Olson[30] , a research engineer at the Jet Propulsion Laboratory(JPL), has developed a grid based terrain-reference localization scheme for use on mobile robots that will explore Mars. These robots need some form of self-positioning in order to leave the general area of the lander. The system that JPL has developed utilizes the Hausdorff distance for matching images[31]. Olson calculates the position of the robot in two steps. First, he generates a most likely location for the robot. Then using a Kalman filter, he integrates his new estimated location with all pervious locations calculated to generate an optimal location.

For a maximum likelihood estimate to be derived, the position of the robot must be written as a function of the map. Olson achieves this by formulating the position of the robot as the distance from the visible features. He compares this to the distance to the closest features in the global map. These are combined by referring the local features to the robots position (x) and referencing the global

0990706.051101

features to x as well. The minimum distance between the local and global features referenced to position x then becomes:

$$D_i^X = \min_{i \le j \le m} d_{ij}^X \tag{1.1}$$

Although these pairs are not independent, Olson has found that little accuracy is lost by assuming that they are independent. The likelihood measure is then found by multiplying the probabilities of these distances with the probability that the robot is at point x. For convince he takes the natural log of the likelihood equation and yielding his final measure:

$$\ln L(X) = \ln p(X) + \sum_{i=1}^{n} \ln p(D_i^X) \tag{1.2}$$

In order to determine the robots location, he uses a branch and bound technique that discretizes the feature locations into a certainty map. This map is then used in conjunction with grid based map matching methods to determine the most likely location for the robot.

To help in constraining the robots location, Olson uses an extended Kalman filter to merge the most likely location with the dead-reckoning location estimate. He has been able to determine the necessary covariance matrixes required to build this filter. This filter smoothes the path and helps to cull outliers from his position estimates. This system is an example the current research in grid based positioning.

Feature Based

Feature based terrain matching uses easily identifiable features such as corners, planes, and edges in indoor environments and builds a metrically accurate map of

these landmarks[32, 33, 34, 35]. Smith, Self and Cheeseman[35] produced one of the first successful implementation of this concept with their work in stochastic mapping. This positioning method tracks the state of the vehicle and the position of all the landmarks in one single state vector and also computes an estimated error covariance matrix. The system is formulated in terms of an extended Kalman filter [36, 37]. The system integrates the input from the sensors (typically a sonar or laser range finder) and vessel dead-reckoning to produce an estimated location of the vessel. Localization is performed by updating of the state vector in a Kalman filter. This approach assumes that all the errors in the system are Gaussian. The successful implementation of feature based navigation requires that significant features can be extracted from both the local and global maps. It also assumes that the errors in matching the features between the maps are few. This method also has a computational complexity problem. This is due to the fact that as the area of interest grows so does the number of vehicle to feature and feature to feature correlations that must be computed and maintained.

Feder [38] developed a system to concurrently map and navigate in the environment using feature based positioning. Although his system deals with both the mapping problem as well as the localization problem, this document only deals with the localization problem so it only summarizes the parts of his system that deals with the localization problem. Feder's system works in two stages. First, he predicts a location of the vessel based on dead-reckoning. Then he uses the re-observation of the key features to perform the localization.

In order to predict the location of the vessel at the next discrete time estimate, he develops a system state model. The model he uses is based on Newtonian physics and contains four states. These states are north position, east position, forward velocity and heading. The prediction step then consists

of projecting the north and positions forward in the heading direction by the velocity scaled to the time increment. This is a simple dead-reckoning position estimate. This position assumes that there are no unknown forces acting on the vessel and the heading and velocity measurement are perfect. These assumptions are obviously incorrect and so the position estimate is refined by re-observing the key features.

The position estimate is refined through the update step in the extended Kalman filter. He estimates by means of a complex process, the covariance of the feature correlation and predicts his location by adding a portion of the difference between the observation and the prediction in accordance with the Kalman gain. Due to the large number of uncertainties in the system he precomputes the Kalman gain and holds it as a constant during vehicle operations. The output of his Kalman filter then functions as the output for his system.

Topological Methods

The final approach to terrain based positioning discussed herein is topological methods. These methods attempt to mimic the means that humans use to locate themselves. Topological localization does not use metric based maps, but rather builds graph like maps where nodes correspond to "significant areas" in the environment. These areas are easy to distinguish and the lines correspond to arcs that are the natural connection between the nodes. Such approaches have been utilized by several groups of researchers [39, 40, 41, 42, 43, 44, 45]. These methods have the disadvantage that they do not use a metric to determine the position.

Lucido et al.[11] have implemented a topologically based positioning system. Their system consists of three steps. First, they create cliff maps from both

09905706 .051101

the global and local maps. Then, they register the two graphs to find the best correlation between the images. Finally, they extract the translation to improve their position estimate for the underwater vessel.

The first step is the generation of cliff maps. Cliff maps are graphs of steep gradients extracted from bathymetric data[46]. These cliff maps are constructed in two stages, convolution of the bathymetry with a Laplacian-of-Gaussian filter and the detection of zero crossings. The Laplacian extracts areas of sharp contrast from the image and the Gaussian helps to remove noise.

After cliff maps (graphs) are extracted from the bathymetric data, critical points are determined for each graph. These points are defined to be places along the contours where the curvature exceeds a certain threshold. The points are then rendered to each other by comparing depth, gradient and the curvature values. The degree of match is then reduced by the square of the Euclidian distance between the points.

The final stage is to define a transformation from the local map to the global map. The rendering matches the points pairwise and from the pairwise matches a global transformation is generated. Lucidio uses a least squares estimate to define the transformation. In order to achieve optimality, the least squares estimation is performed in an extended Kalman Filter.

1.4 Overview of the Dissertation

This dissertation develops and demonstrates a new accurate underwater positioning system that combines parts of dead-reckoning, acoustic-based, and terrain-based positioning into a single integrated system. The system requires a detailed, high resolution bathymetric map in the operational area, the slant range to a known position and the depth of the ocean at the position of the body. These inputs are

0990706.051101

then combined to generate an accurate estimate of the position of the underwater vessel. Typical system setups are included for towbody operations, figure 1.5 and AUV operations, figure 1.6.

This document is organized as follows. Chapter 2 details the model of the system and the overall system framework. Chapter 3 contains a detailed explanation of the terrain matching algorithm used for the system. Chapter 4 details the steady state Kalman filter used to integrate the terrain matching with dead-reckoning. Chapter 5 explains the system software used for the LOST2 system. Chapter 6 explains the results of computer simulations. Chapter 7 presents the real data that was obtained for system validation. Chapter 8 presents the results of real world system testing. Chapter 9 concludes with overall conclusions and suggestions for further work.

09905706.051101

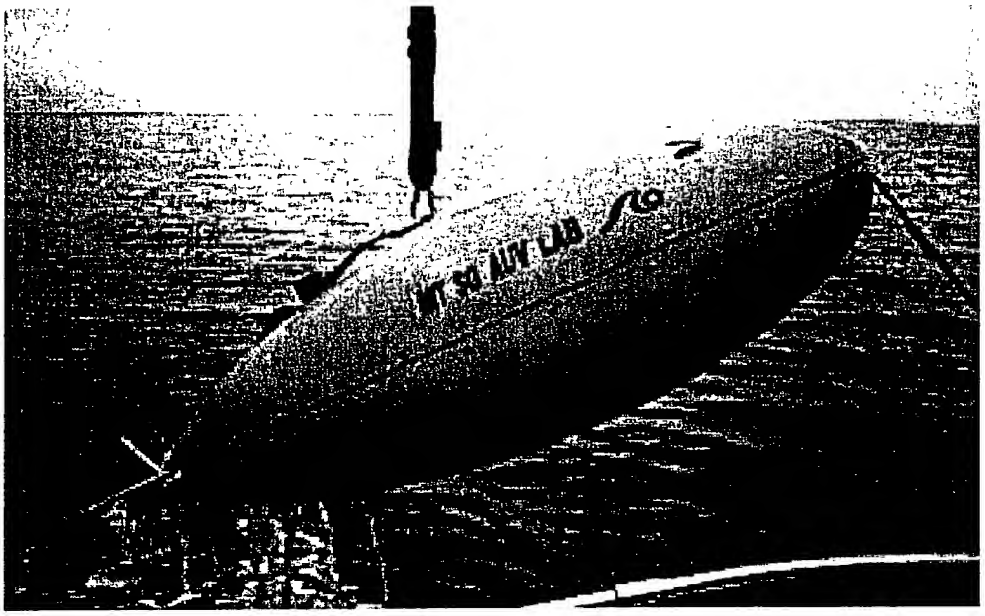

Figure 1.1: Odyssey IIB AUV

0990570606.051101

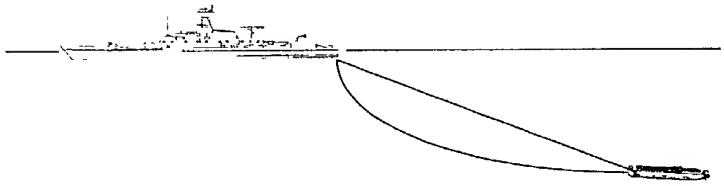

Figure 1.2: Cable Layback Method,Cable is straight

09905706.051101

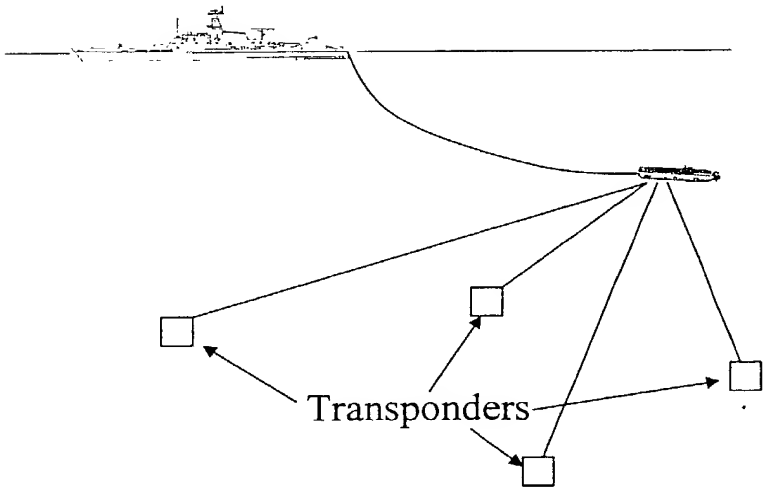

Figure 1.3: Long Baseline Method

09905706.051101

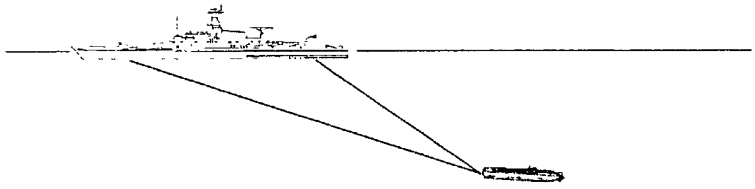

Figure 1.4: Short Baseline Method

0990706 .051101

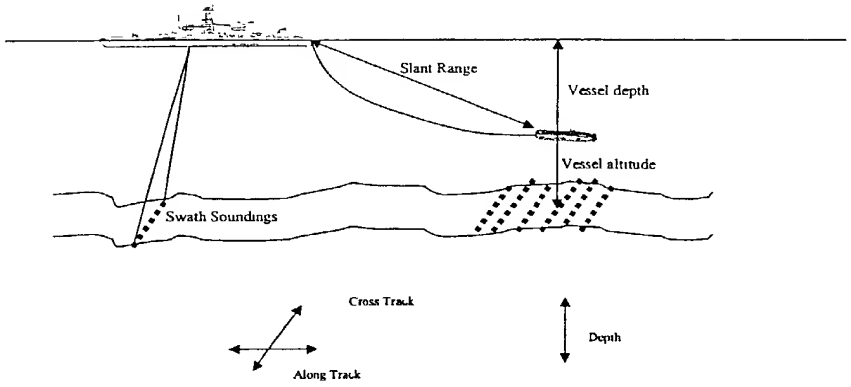

Figure 1.5: Example Implementation for Towbodies

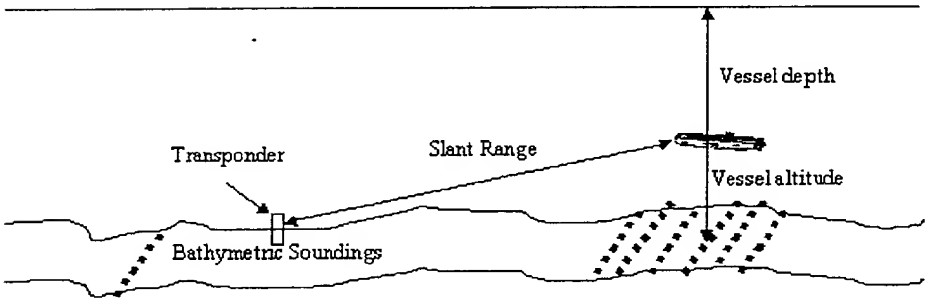

Figure 1.6: Example Implementation for AUVs

0990706.051101

Chapter 2

Overall System Description and the Linear System Model

The LOST2 underwater positioning system combines parts of dead-reckoning, acoustic-based, and terrain-based positioning into a single integrated system. The system is designed in two subsystems, the system observer and the constrained extended Kalman filter.

2.1 Overview of the function of each subsystem

This section describes the functionality as well as the input and output of each of the subsystems in the LOST2 system. The system is designed in two subsystems shown in figure 2.1, the system observer and the constrained extended Kalman filter. The system observer subsystem is composed of three modules: a state velocity update, terrain matching module, and a spatial performance prediction. The constrained extended Kalman filter subsystem consists of three modules: a steady state extended Kalman filter, a non-linear constraint module, and a state predictor. The system observer integrates the high resolution bathymetry, vessel's measured ocean depth, vessel's predicted state, the vessel's measured velocity, and

0990706.051101

the measured slant range and location of the known point into the terrain based state estimate. This system outputs the position estimate, a final predicted state and the measured slant range and location of the known point. The system observer also computes a spatially based prediction of system performance. The constrained extended Kalman filter takes the terrain based state estimate, a final predicted state, the measured slant range and location of the known point and computes the final estimate of state and a prediction of the next system state.

The inputs that the system requires are: a detailed, high resolution bathymetric map in the operational area, a slant range to a known position, an estimate of the body's state, a measured or estimated velocity and the depth of the ocean at the position of the body. These inputs are used in conjunction with the output of the other modules to generate an accurate estimate of the position of the underwater vessel.

2.1.1 The System Observer

The first subsystem is the system observer, which is included as figure 2.2. This subsystem's primary function is to generate all the predictions of the state that is required by the constrained extended Kalman filter. It also provides a prediction of performance based on the spatial distribution of the bathymetry. This subsystem integrates the high resolution bathymetry, vessel's measured ocean depth, vessel's predicted state, the vessel's measured velocity, and the measured slant range and location of the known point into the terrain based state estimate. The subsystems outputs are the terrain based state estimate, the final predicted state and the measured slant range and location of the known point.

The first subsystem component is the state velocity update, whose signal flow diagram is contained in figure 2.3. This module serves to provide an estimated

09905706 .051101

location of the underwater body. It does this by combining the best available velocity (speed and heading) and the best estimate of state to produce the best estimate state prior to the terrain match. This module then outputs the predicted state at the present time.

The output of the state velocity update is sent to the terrain matching module, whose signal flow is shown as figure 2.4. This part of the system serves as the independent system observation. The observation is made by comparing the measured ocean depth to the bathymetry in the operational area. This is done by means of a maximum likelihood estimator. The output of this module is the terrain based position estimate, which is also the independent system observation.

The final module of the system observer is the spatial based performance prediction. The signal flow is contained in 2.5. This module, based on the high resolution bathymetry in the area, produces an estimate of how well the system is capable of performing in a given area.

2.1.2 Constrained Extended Kalman Filter

The second subsystem of the LOST2 system is the constrained extended Kalman filter, whose state flow diagram is contained in figure 2.6. This subsystem builds on Kalman filter theory, extended to nonlinear systems, to integrate all of the state estimates into a single final state estimate. This subsystem takes as input the terrain based state estimate, a final predicted state, the measured slant range and location of the known point and computes the final estimate of state and a prediction of the next system state.

The steady state Kalman filter, contained in figure 2.7, is the heart of the constrained extended Kalman filter. This filter merges the two estimates of state

0990706.051101

is accordance with Kalman filter theory. This module takes as input the two estimates of state and outputs a linear combination of the two.

The next module forms the constrained part of the constrained extended Kalman filter. A signal flow diagram is shown in figure 2.8. Due to the errors in linearization and the lack of a good system model, the linear filter's output must be constrained. This is done in two stages. First, the amount of change from the predicted state is thresholded. Secondly, the slant range is used to force the estimated location to be the correct distance from the known point.

The final module in the constrained extended Kalman filter is the state predictor, which is contained in figure 2.9. This module projects the state ahead by keeping an internal running estimate of the unmeasured velocities and adds this to the final estimated state. This is the first stage in preparing one of the state estimates for the next time step.

2.2 The Linear System Model

Before any development can be accomplished, the mathematical representation of the system must be determined. It was determined that the mathematical model would be in the form of a state space representation as this is consistent with both control and observation theory[47]. This section develops the state space model that is used to form the basis of development for all the system modules.

In order to simplify the model, it was decided to only model the body's two dimensional position. This was made for two reasons. First, it reduced the number of states that have to be computed, reducing the computational complexity of the system. Secondly, the vessel's vertical position is assumed to be measured to a high degree of accuracy. This assumption is because the system requires the depth

09905706.051101

of the water to be known accurately. Therefore, only the vessel's planar motion is modeled.

The first step in developing the model was to choose the states of the system. A common state representation is to model the state by the north position, east position, heading and speed[38]. However, an equivalent set of continuous time states exists in the form of the body's north position, north velocity component, the body's east position and the east component of velocity. In discrete time, these states become the current north position, previous north position, current east location, and previous east location. During experimentation, it was observed that if instead of using north and east as the basis for the state space, the system tends to decouple if the basis is instead the cross track and along track directions. This can be obtained by applying a rotation matrix R to the state. This yields the final state representation that was used for the system, X.

The general linear model of any system in state space form takes the form of the continuous time equations 2.1 and 2.2.

$$x(t) = Ax(t-1) + Bu(t) \tag{2.1}$$

$$y(t) = Cx(t) + Du(t) \tag{2.2}$$

Where A is the state transition matrix, B is the control matrix, u is the control input, y is the observation of the system, C is the observation matrix and D is a mapping from the input space directly to the output space[47]. Due to the nature of the system being an observer only, the value of u is only partially known and not changeable by the system. It is therefore split into two pieces, the known (measured) part and the unknown part. This has the effect of adding a noise process to both equations 2.1 and 2.2. The noise is assumed to be random zero

0990706.051101

mean and for ease of calculation it is further assumed that the noise can be modeled by white noise processes v_k and w_k respectively. Substituting the white noise processes and discretizing yields the form of the final linear discrete model used.

$$x_{n+1} = Ax_n + Bu_n + w_n \tag{2.3}$$

$$y_n| = Cx_n + Du_n + v_n \tag{2.4}$$

Due to the choice of state variables described above, the full four state model can be broken into two independent models each representing one of the two orthogonal directions. The underwater vessel's dynamics in each direction are assumed to obey standard Newtonian motion. There are two forces assumed to be acting on the body, a frictional force F_c and an unknown outside disturbing force $u(t)$. The continuous time differential equation for a body with mass m and position $p(t)$ is then:

$$m\ddot{p}(t) = -F_c\dot{p}(t) + u(t) \tag{2.5}$$

Using the position, $p(t)$, and velocity, $\dot{p}(t)$ as the states of the system,

$$x(t) = \begin{bmatrix} p(t) \\ \dot{p}(t) \end{bmatrix}$$

produces the continuous state space representation:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{F_c}{m} \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) + w(t) \tag{2.6}$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u(t) + v(t) \tag{2.7}$$

Note that the position is assumed to be the only observable element of state.

Discretizing the system using a difference approximation to the derivative, and defining $\mu = \frac{F_c T}{m}$ yields the discrete form of the state space representation with new state vector

$$x = \begin{bmatrix} p_n \\ p_{n-1} \end{bmatrix}$$

and state equations

$$x_{n+1} = \begin{bmatrix} 2-\mu & \mu-1 \\ 1 & 0 \end{bmatrix} x_n + \begin{bmatrix} T^2/m \\ 0 \end{bmatrix} u_n + \begin{bmatrix} 1 \\ 0 \end{bmatrix} w_n \tag{2.8}$$

$$y_n = \begin{bmatrix} 1 & 0 \end{bmatrix} x_n + v_n \tag{2.9}$$

where p_n is the position at time n. Repeating this process for both directions yields the final system model used for the system, this is shown as equation 2.10.

$$x_{n+1} = \begin{bmatrix} 2-\mu_a & \mu_a-1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 2-\mu_c & \mu_c-1 \\ 0 & 0 & 1 & 0 \end{bmatrix} x_n + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_n + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} w_n$$

$$y_n = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} x_n + v_n \tag{2.10}$$

This model will be used to develop the Kalman filter in chapter 4.

0990706 051101

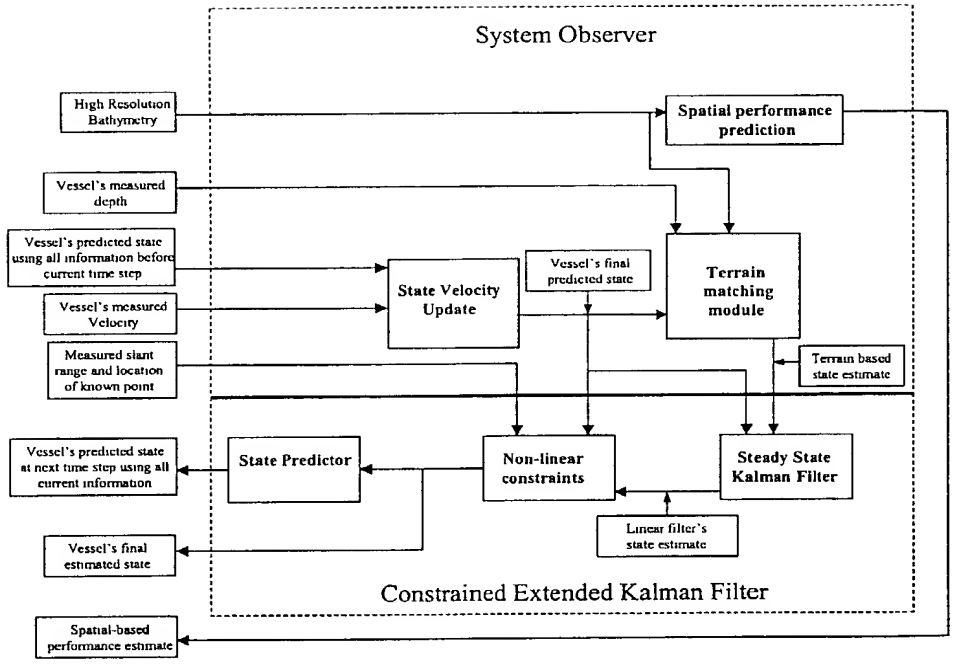

Figure 2.1: Block diagram of the system showing the two primary subsystems

0990706 .051101

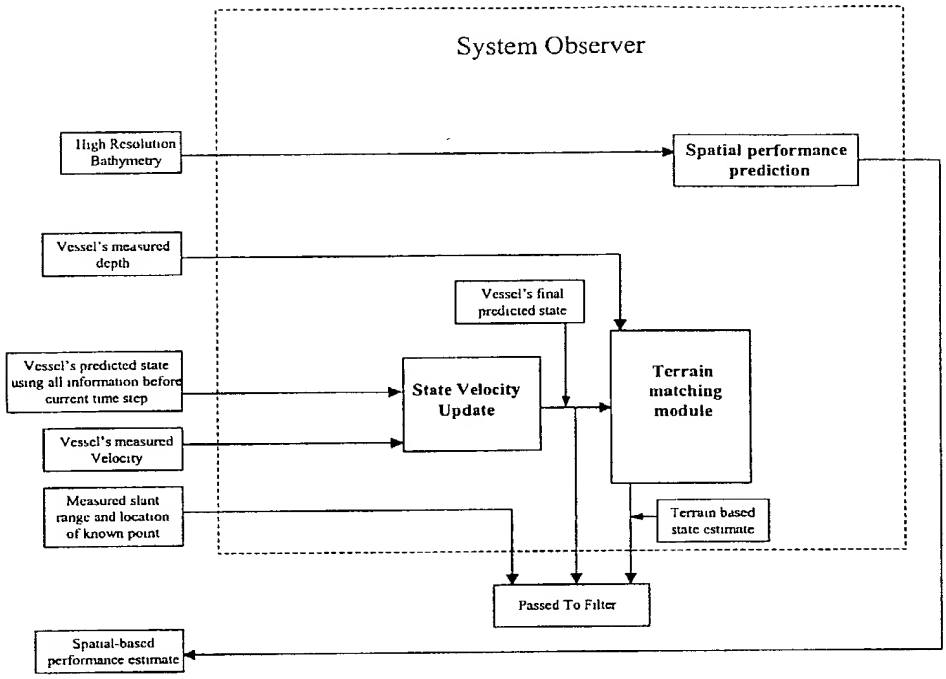

Figure 2.2: Block diagram of the system observer subsystem showing the signal flow

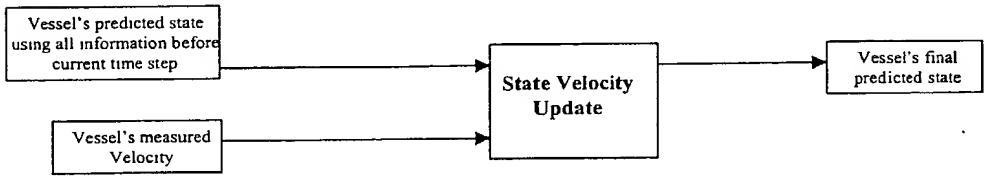

Figure 2.3: Input/Output diagram for the state velocity update

0990570606.051101

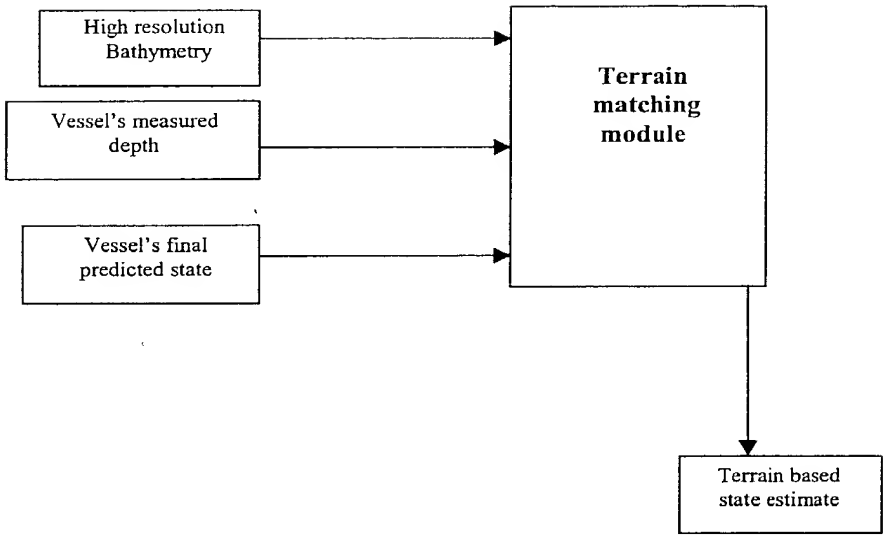

Figure 2.4: Input/Output diagram for the terrain matching module

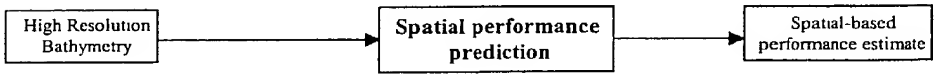

Figure 2.5: Input/Output diagram for the spatial based performance prediction module

09905706 .051103

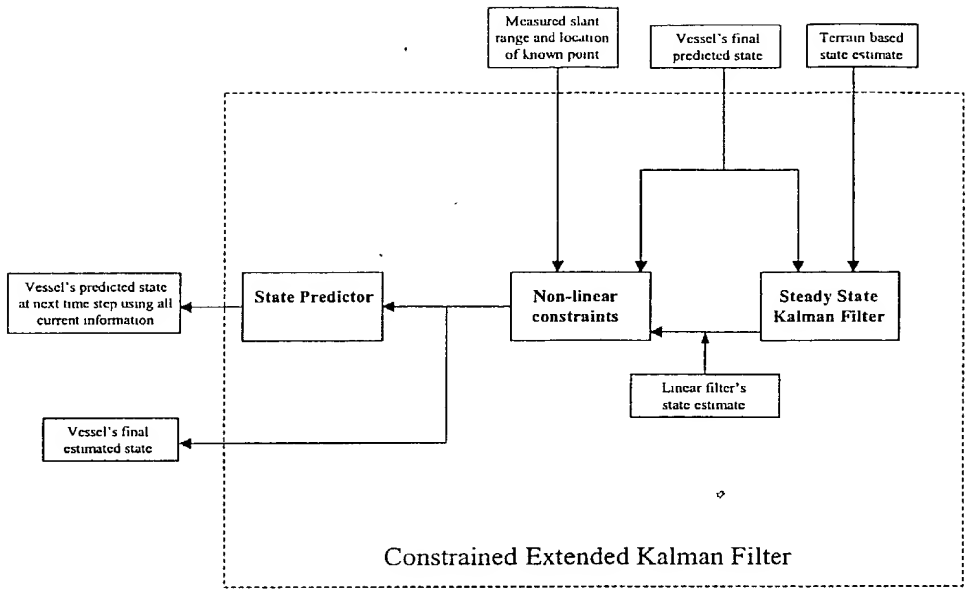

Figure 2.6: Block diagram of the constrained extended Kalman filter subsystem showing the signal flow

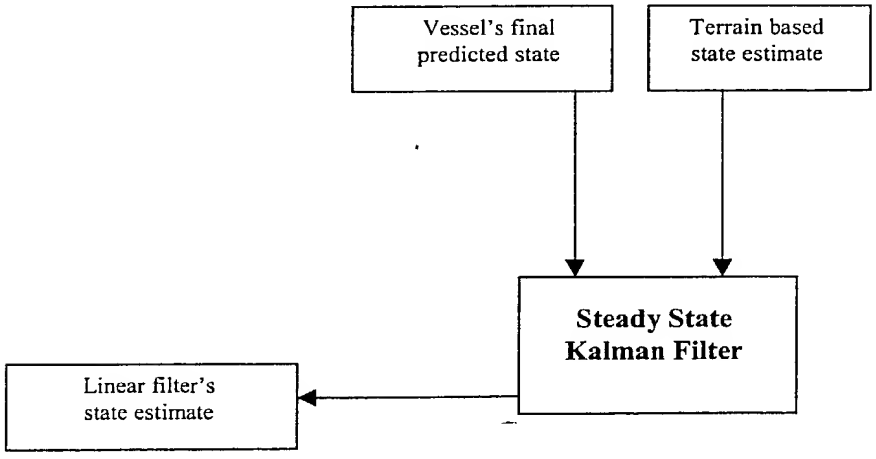

Figure 2.7: Input/Output diagram for the steady state Kalman filter

09905706 .051101

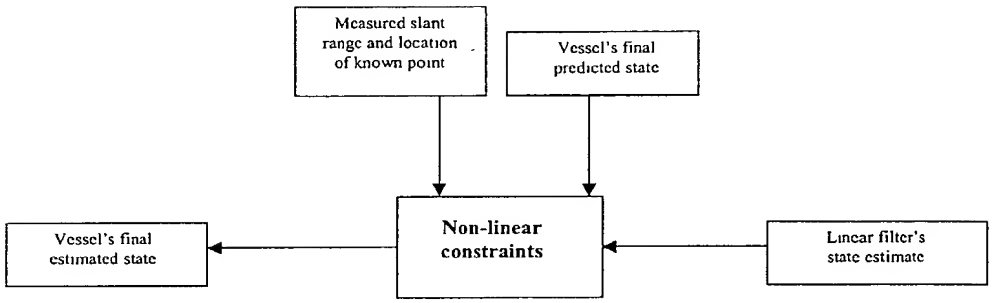

Figure 2.8: Input/Output diagram for the nonlinear constraint module

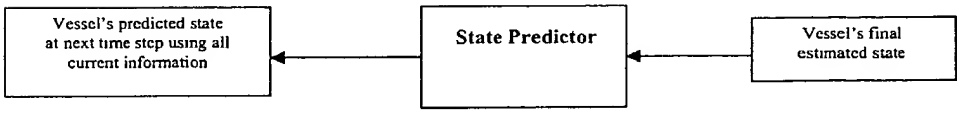

Figure 2.9: Input/Output diagram for the state predictor

0990706.051101

Chapter 3

System Observer

The LOST2 system's first major subsystem is the system observer. This subsystem takes in all the available measurements and generates two position estimates, one from Newtonian motion and the other based on the terrain in the area. This subsystem also contains some of the work done in conjunction with Leslie Morgan, which provides an estimate of the best performance based on the nature of the terrain over which the LOST2 system is operating.

This chapter is divided into three sections. The first section describes the theory used to generate the terrain-based position estimate. The second section describes the two modules that generate the position estimates which are passed into the constrained extended Kalman filter described in chapter 4. The last section contains the means to estimate the performance of the system.

3.1 Point Estimation Theory

In order to generate a position estimate from the measured depth and the bathymetric data, a means to chose the best estimate of location must be determined. Given a distribution of the data with some unknown parameter, methods have been

derived to provide a single value that is the "best" approximation of the unknown parameters. This is known as the point estimation problem.

Given any unknown parameterized probability density function (pdf), assume that the family of distributions it belongs to is known, but at least some of the parameters are unknown. Then it becomes useful given some observations from the distribution, to develop a method to estimate the unknown parameters. This method results in evaluating different possible forms of the pdf to find the one that best describes the observations.

Therefore, we define the likelihood function $L(\theta|y)$ as $f(y|\theta)$ and the log likelihood as the natural log of the likelihood function[48]. Simply restated the likelihood function corresponding to the observed vector y from a distribution $f(y|\theta)$ is written as

$$L(\theta|y) = f(y|\theta) \tag{3.1}$$

Then a maximum likelihood estimate (mle) of the parameter can be determined. The value of this estimate of the parameter is then the $\hat{\theta}$ that maximizes $L(\theta|y)$ or its logarithm $l(\theta|y)$[49].

The most practical means to determine the mle estimate is to find and examine the local maxima and minama of the likelihood function. The mle estimate of the parameter in question, in the logarithm case, then becomes the solution to the likelihood equation given by

$$\frac{\partial l(\theta|y)}{\partial(\theta_s)} = 0 \quad (s = 1, ..., n) \tag{3.2}$$

The solution to 3.2 that maximizes 3.1 becomes the final estimate of the parameters. This method forms a good means to estimate the parameters when the correct

09905706 .051101

solution is not available. A more comprehensive derivation of this theory can be found in [48] and [49].

3.2 System Observer position estimates

The System Observer generates two separate estimates of the position of the underwater body. The first position estimate is generated in the velocity update. This position estimate is then used in conjunction with bathymetry to generate the terrain based position estimate.

3.2.1 Velocity Update

The velocity update serves as the dead reckoning system. This module computes an estimated position for the vessel in the along track, cross track coordinate system based on a measured velocity vector for the underwater vessel, and the best prediction of position of the underwater vessel from the previous time step.

The state predictor calculates the new position (along track, cross track) as two separate and independent processes, one in each direction. This is done through out the system. Because the coordinate system is orthogonal, combining the results into a single position is trivial.

For each direction the position estimate is computed in by adding the measured velocity, multiplied by time as appropriate, to the previous position estimate. The mathematics to compute the estimate are:

$$\hat{x}_n^a = x_{n|n-1}^a + v_n^a \tag{3.3}$$

$$\hat{x}_n^c = x_{n|n-1}^c + v_n^c \tag{3.4}$$

where $x_{n|n-1}$ is the best estimate of the position before the time n measurements, v_n is the velocity times the time between samples at time n and a superscript a stands for the along track direction and a superscript c stands for the cross track direction.

The resulting prediction of position is provided to both the terrain matching module, and the constrained Extend Kalman filter. This prediction is used by the terrain matching module to generate the second observation of position.

3.2.2 Terrain Based Position Estimate

The terrain matching module takes as its inputs the position generated by the velocity update module, the measured ocean depth at the vessel's position and the high resolution bathymetry for the area of operation. This module then outputs the its best estimate of position based on mle theory.

In order for useful observations to be made, a reasonably dense, spatially varying bathymetric data set in the neighborhood of the vessel is required. Each point in this bathymetric data set is a triple, $\alpha_i = (x_i, y_i, z_i)$. Where x_i, y_i is the 2-D position vector (eastings and northings) in meters and z_i is the measured depth in meters. In practice, this data can be collected previously or concurrently.

Given the dead reckoning estimate of vessel position $\hat{x}_n$ and the measured ocean depth at the vessel's position, z_n, the 3D position vector, $\alpha_n^- = (\hat{x}_n, z_n)$, can be formed using the same units as α_i after rotating coordinate systems. Without loss of generality, it is assumed that the ocean bottom depth is measured directly beneath the vessel to simplify this exposition; in practice it may be necessary to correct for vessel orientation, which is possible through a set of rotations. If the true location of the vessel and ocean bottom depth is denoted as $\alpha_n = (x, y, z)$,

then the measurement error, defined as the difference between the true and dead reckoned positions, can be written as $e = \alpha_n - \alpha_n^-$.

Viewed as an estimation problem, a maximum likelihood approach is used. That is, given several observations from a distribution with unknown parameters the value of the observation that maximizes the likelihood function is the best estimate of the parameters. The α_n and α_n^- are related through the likelihood function(LF) based on a known or assumed error distribution, f_e. The LF can be expressed as

$$L(\alpha_i|\alpha_n^-) = f_e(\alpha_i - \alpha_n^-) \qquad (3.5)$$

Given the estimated position α^- and the bathymetry in the area $\{\alpha_i\}$ the most likely location of the vessel constrained to be in the bathymetric data set can be determined as that α_i which maximizes $L(\alpha_i|\alpha^-)$. When applied to the bathymetric data set $\{\alpha_i\}$, (3.5) produces a measure which can be viewed as the weighted distance from the estimated position, $\hat{x}_n$. Selecting the bathymetry triple which minimizes this distance is the maximum likelihood estimate of position constrained to the bathymetric data set.

If the error e is assumed to be zero mean Gaussian with covariance Σ, then the likelihood function in (3.5) is

$$L(\alpha_i|\alpha_n^-) = f_e(\alpha_i - \alpha_n^-) = |2\pi\Sigma|^{-1/2} \exp\left(-0.5(\alpha_i - \alpha_n^-)^T\Sigma^{-1}(\alpha_i - \alpha_n^-)\right) \qquad (3.6)$$

Maximizing this equation with respect to α_i is readily shown to be equivalent to minimizing

$$\lambda(\alpha_i|\alpha_n^-) = (\alpha_i - \alpha_n^-)^T\Sigma^{-1}(\alpha_i - \alpha_n^-) \qquad (3.7)$$

Because Σ is a covariance matrix, it is positive semidefinite, the likelihood function in (3.7) is strictly non-negative with minimum 0 at $\alpha_i = \alpha_n^-$.

0990706.051101

The correct value for the covariance matrix depends on the nature of the bathymetric data being used. The covariance matrix Σ is given by:

$$\Sigma = \begin{bmatrix} \sigma_a^2 & 0 & 0 \\ 0 & \sigma_c^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix} \tag{3.8}$$

where σ_a^2 is the variance in the along track direction, σ_c^2 is the variance in the cross track direction, and σ_z^2 is the variance in the z direction. Substituting this matrix into 3.7 and performing the indicated minimization generates the mle's observation of position which is then passed to the constrained extended Kalman filter.

3.3 Best Performance Estimation

This work was done primarily by Leslie Morgan, in conjunction with the LOST2 system development. A parameterized model for the distribution of the point-to-event distances within the multibeam bathymetry data is desired, whose expected value the estimate of best performance. This parameterized model is needed to estimate the lower limit of the positioning error for the LOST2 system being developed. The positioning error is a function of the density and spacing of the bathymetric data. The parameterized distribution should have a lower limit of zero since there will be no negative distances from an arbitrary point to the closest bathymetric point. The data should be transformed from linear distances to circular areas. Transforming the data in this manner is logical because as one moves outward from an arbitrary point in search of the nearest point, one is moving outward along the radius of a circle.

The data set that was used to develop the parameterized model was obtained from Pensacola Bay, Florida. The data is not gridded and contains (x, y, z) coordinate points in meters. The analysis done on this data focused on interior regions of the dataset to avoid edge effects. This is a valid restriction in that terrain-based navigation systems should not be used on the edge of available bathymetry data.

Five empirical cumulative distribution functions (cdf) for the point-to-event distances were used to determine an appropriate parametric distribution. It is desirable for one distributional type to be appropriate for the entire multibeam data set. The estimated parameters may change in different regions, but the distributional type should not.

Six standard survival distributions were selected as potential candidates. The models selected were the generalized gamma, the exponential, the Weibull, the standard gamma, the log-normal, and the log-logistic distributions. Typically survival distributions are used to estimate the time-to-event. By examining the log-likelihoods, it was shown that the generalized gamma provides the best fit followed by the Weibull distribution.

The generalized gamma is a three parameter distribution involving the gamma function and the incomplete gamma function. The exponential, Weibull, standard gamma, and log-normal models are all special cases of the generalized gamma distribution. The third parameter of the generalized gamma allows its hazard function to take on a wide variety of shapes. The generalized gamma distribution will fit unless the hazard function has more than one peak. However, if one of the simpler models can be shown to fit, the generalized gamma is not used for three main reasons. First, the pdf is complicated and the parameters are difficult to interpret. Second, the computer time to estimate the generalized

09905706.051101

gamma is significantly longer than for the simpler models, and third, the generalized gamma has a reputation for convergence problems[51]. For these reasons, the Weibull distribution was selected as the potential model.

The Weibull model is a slight modification of the exponential model, with the important consequence that the hazard rate is no longer constant. The Weibull cdf incorporating the transformation to radial areas is given by

$$G(r) = 1 - \exp(-\lambda(\pi r^2)^\gamma)$$

with $r \geq 0, \lambda > 0, \text{and} \gamma > 0$. λ is a scale parameter and γ is a shape parameter. When $\gamma < 1$ the Weibull distribution has a decreasing hazard rate. When $\gamma = 1$, the Weibull cdf reduces to the exponential. When $1 < \gamma < 2$, the hazard rate is increasing at a decreasing rate. When $\gamma = 2$, the hazard function is an increasing straight line with origin at zero. When $\gamma > 2$, the hazard function is increasing at an increasing rate.

The method of maximum likelihood was used to estimate the parameters for a Weibull distribution for the five empirical cdf's. The expected value of the Weibull distribution is the estimated best performance of the terrain matching. A much more detailed explanation is available in [52].

19980706 051101

Chapter 4

The Constrained Extended Kalman Filter

Three of the modules of the LOST2 system form a constrained extended Kalman filter. This chapter provides an overview of Kalman Filter theory, and extends this theory to nonlinear systems. Then, the extended Kalman filter that the system uses is derived, including the addition of nonlinear constraints to improve stability.

4.1 Kalman Filter Theory

A Kalman filter is the optimal linear mean squared error estimator for linear dynamic systems. These filters exist in both continuous and discrete time versions, However, only discrete time versions are discussed here as this system is assumed to be implemented on a computer. The linear version of the filter is discussed first, and then the extensions to nonlinear systems are developed.

09905706.051101

4.1.1 Linear Kalman Filter

Given a linear dynamic system, the Kalman filter is an efficient linear minimum means squared error estimator[53, 37, 54, 55]. In addition if the noise processes present in the system are Gaussian in nature, the Kalman filter is the optimal estimator of state. If the noises are not Gaussian, the Kalman filter is the best linear estimator. The Kalman filter consists of four steps. These are prediction, observation, update and then repeat.

The Kalman filter operates on a state space representation of a stochastic system. The discrete time state space representation can be written in vector form as

$$x_{n+1} = Ax_n + Bu_n + w_n \tag{4.1}$$

where x_n is the stochastic system's true state at time n. The matrix A is the discrete time linear system model, B is the control matrix, u_n is the control input at time n, and w_n is the discrete time Gaussian noise process.

In addition, it is assumed that there exists an observer to provide a noisy estimate, z_n of some of the state variables. This observation is assumed to be modeled by a linear observation matrix, H, and a mapping matrix D such that observation equation becomes

$$z_n = Hx_n + Du_n + v_n \tag{4.2}$$

where v_n is the observation noise process.

09905706.051101

The noise process are further assumed to be white, uncorrelated Markov processes [54] with covariances G and R respectively. That is the system covariance, G, and the observation covariance, R, are given by

$$E(v_n, v_k^T) = \delta_{nk} G \tag{4.3}$$

$$E(w_n, w_k^T) = \delta_{nk} R \tag{4.4}$$

and

$$E(v_n, w_n^T) = 0 \tag{4.5}$$

with

$$\delta_{nk} = \begin{cases} 1 & \text{if } k = n \\ 0 & \text{if } k \neq n \end{cases} \tag{4.6}$$

It can be further assumed that the noise processes have zero mean.

For notational purposes, let x_n denote the system's true state at time n. Further, let $x_{n|n-1}$ represent the estimated state at time n given all information up to but not including time n, and $x_{n|n}$ represent the estimate after the time n observations have been integrated. The error is then defined as

$$\tilde{x}_{n|n} = x_{n|n} - x_n \tag{4.7}$$

Using these definitions, the Kalman Filter can be formulated as a linear minimum mean squared error estimator. The final estimate of a Kalman filter after the observation is made, $x_{n|n}$, is found as follows

$$x_{n|n} = x_{n|n-1} + K_n(z_n - H x_{n|n-1}) \tag{4.8}$$

where K_n is the optimal gain, which still has to be determined.

09905706.051101

The value of the optimal filter gain is determined by noting that the Kalman filter is by definition a minimal mean squared error filter. This can be express mathematically as

$$x_{n|n} = \min_{\tilde{x}_{n|n}}(E(\tilde{x}_{n|n}\tilde{x}_{n|n}^T) \tag{4.9}$$

The expected value in the previous formula is the covariance and for notational purposes will be referred to as P. Rearranging and substituting 4.8 into the following equation yields.

$$P_{n|n} = (I - K_n H)P_{n|n-1}(I - K_n H)^T + K_n R K_n^T \tag{4.10}$$

The optimal gain is the one that minimizes the trace of $P_{n|n}$ or

$$\frac{\partial}{\partial K_n}\mathrm{tr}(P_{n|n}) = 0 = -2(I - K_n)P_{n|n-1}H^T + 2K_n R \tag{4.11}$$

and solving for K_n yields the Kalman Gain

$$K_n = P_{n|n-1}H^T(HP_{n|n-1}H^T + R)^{-1} \tag{4.12}$$

Now that the optimal method to integrate the observation has been determined, it is necessary to make a prediction of the state and covariance at the next time step. The state update is done by definition

$$x_{n+1|n} = Ax_{n|n} \tag{4.13}$$

0990706.051101

and the covariance is predicted by

$$P_{n+1|n} = AP_{n|n}A^T + G \tag{4.14}$$

The prediction step can be augmented by a known control input BU_n by adding this to the prediction step.

4.1.2 The Extended Kalman Filter

The fundamental assumption of the Kalman filter is that both the system model and the observation model are linear. Under these conditions, the filter is an optimal estimator. However, most real world systems are not linear and therefore Kalman filter theory does not directly apply. In order to implement Kalman filters on nonlinear systems, extended Kalman filters have been developed.

These filters are derived by linearizing the models. This is done by taking the first two terms of the Taylor series expansion. The assumption in this linerazation is that the errors in estimation are small. The only change in the filter derivations is the replacement of the nonlinear models by their linear equivalent. In general, this methods works when the errors in linearization are small[38].

Another approach to extended Kalman filters is based on the fact that once the filter converges to the optimal gain and covariance estimates that these values tend to remain constant. Therefore, the nonlinear system can be linearized by approximating these steady state values. This is usually done by a minimax approach, or minimizing the total error over a large number of trials.

The extended Kalman filter, no matter it derivation, suffer from two problems. The first the resulting filter is generally suboptimal, due to the approximations involved in the derivation of the filter. Secondly, a spurious reading can cause the filter to diverge, especially if the model is poor.

0990706 .051101

4.2 The Constrained Extended Kalman Filter

Three of the modules in the LOST2 form a constrained extended Kalman Filter. This is a steady state extended Kalman filter with nonlinear constraints. These constraints placed on it to improve performance and stability given the poor nature of all models for underwater vessels[38].

The constrained extended Kalman filter is required for three reasons. First, the true model of the system is highly nonlinear and impossible to model accurately. The second is that the covariance matrix is impossible to estimate. The third reason is that the nature of the system makes it highly prone to outliners, which is corrected for by the nonlinear constraints.

The system model is highly nonlinear and does not linearize easily[38]. Therefore, a simple linear model was developed for the system. The consequence of this is that the model bears only a general correspondence to the real system.

Because the model is imprecise, the true covariance is impossible to predict. Therefore, the Kalman gain and covariances are determined via the minimax procedure describe in the previous section. This results in the filter being sub-optimal, when compared to the filter that could be built if more information was available.

The nature of the observer and the model situation result in a system that is prone to divergence and outliers. To prevent both of these problems, nonlinear constraints were added to the standard extended Kalman filter.

The modules the form the basis of the constrained extended Kalman filter consist of the state predictor, state estimator and the nonlinear constraints module. The state predictor and state estimator form an extended Kalman filter, while the addition of the constraint module forms the constrained extended Kalman filter.

0990570G .051101

4.2.1 The linear extended Kalman filter

After the prediction and observation steps of the Kalman filter are completed, the position estimate of the system observer must be merged. The terrain-based observation (y) is merged into the prediction by the velocity update by means of a linear filter. The linear extended Kalman filter is the merging step of the constrained extended Kalman filter.

This module works in two steps. First the error or innovations, η, of the system are computed by subtracting the observation from the prediction. The second step is to merge the innovations with the prediction via Kalman filter theory.

The innovations are computed in each of the two independent directions separately. The formulation is the same, but for completeness is expressed in the following two equations:

$$\eta_n^a = y_n^a - \hat{(x)}_{n|n-1}^a \tag{4.15}$$

$$\eta_n^c = y_n^c - \hat{(x)}_{n|n-1}^c \tag{4.16}$$

After the innovations are defined, they are merged with the prediction in each direction by the Kalman gain K. This can be done in this manner due to the assumption that the system decouples into two independent systems. Again following the Kalman theory, this is expressed as:

$$\hat{x}_{n|n}^a = \hat{x}_{n|n-1}^a + K_n^a \eta_n^a \tag{4.17}$$

$$\hat{x}_{n|n}^c = \hat{x}_{n|n-1}^c + K_n^c \eta_n^c \tag{4.18}$$

This module would in a normal extended Kalman filter complete one cycle through the filter algorithm. However due to the unique nature of underwater

vessels and this system, nonlinear constraints have been added to improve the system.

4.2.2 The Nonlinear Constraints

This filter has two problems associated with it, each of which has been addressed by separate set of nonlinear constraints. The first set of constraints serves to cull outliers from the system by means of a threshold. This threshold limits the distance that the system is allowed to change the predicted location. The second constraint is a slant range from a known point. This serves to aid in convergence and to prevent divergence.

The first set of constraints serves to cull the outliers from the observation module, which tends to cause unpredictable performance. This is accomplished by comparing the difference between $\hat{x}_{n|n-1}$ and $\hat{x}_{n|n}$ in each direction and limiting its magnitude to a certain value ϵ. The effect of this is:

$$\hat{x}^a_{n|n} = \begin{cases} \hat{x}^a_{n|n-1} - \epsilon & \\ \hat{x}^a_{n|n} & \text{when} \quad |\hat{x}^a_{n|n} - \hat{x}^a_{n|n-1}| \leq \epsilon \\ -\hat{x}^a_{n|n-1}\epsilon & \end{cases} \qquad (4.19)$$

$$\hat{x}^c_{n|n} = \begin{cases} \hat{x}^c_{n|n-1} - \epsilon & \\ \hat{x}^c_{n|n} & \text{when} \quad |\hat{x}^c_{n|n} - \hat{x}^c_{n|n-1}| \leq \epsilon \\ -\hat{x}^c_{n|n-1}\epsilon & \end{cases} \qquad (4.20)$$

These limits have the effect of trading convergence time for a vast increase in stability. The unit ϵ is meters and a rule of thumb for a nominal value is 0.1 meters per second (scale this by the sampling rate).

The other set of system constraints is the slant range correction. This corrects the estimated location of the vessel by forcing it to have the correct

D9905706 .051101

horizontal range (magnitude only) from the transponder. It does this by forming a line starting from the transponder and ending at the estimated vessel position. This line is then scaled by moving the estimated position along the line until the range is correct. This serves to increases convergence time by allowing for large movements in position and prevents divergence by preventing positions that are not realistic given the slant range information.

The nonlinear constraints distinguish this filter from other Kalman filters and are an extension off extended Kalman filters dictated by the nature of the system. By introducing nonlinearities into the linear system model, the system predicts the real dynamics better. The output of this module completes the merging of new information into the predicted location and is the best estimate as to the vessel's position.

4.2.3 The State Predictor

The state predictor module serves as part of the prediction stage of the Kalman filter. This modules computes an estimated position for the vessel in the along track, cross track coordinate system based on previous position estimates (stored internally).

The state predictor calculates the new position (along track, cross track) as two separate and independent processes, one in each direction. This is done though out this system. Because the coordinate system is orthogonal, combining the results into a single position is trivial.

For each direction the position estimate is computed in by adding a portion of the unmeasured estimated velocity in each direction. This is done by differencing the previous two position estimates and multiplying the result by $1-\mu$. The value

09905706 .051101

of μ is different in each direction and for each specific system, but a ballpark figure for the correct value is 0.1. The mathematics to compute the estimate are:

$$\hat{x}^a_{n+1|n} = x^a_{n-1} + (1-\mu_1)(x^a_{n-1} - x^a_{n-2}) \quad (4.21)$$

$$\hat{x}^c_{n+1|n} = x^c_{n-1} + (1-\mu_2)(x^c_{n-1} - x^c_{n-2}) \quad (4.22)$$

The resulting position estimate is then fed forward as one of the required inputs of the system observer.

09905706 .051101

Chapter 5

System Software

This chapter contains the details for the software used in the implementation of the system. The code is divided into a main program and six subroutines all coded in the C programming language. The software requires three input files, which as of now are hardcoded into the code, and outputs a file containing the position of the underwater body. The chapter is divided into two sections. The first section briefly describes the function of and input/output of each block. The section contains the actual C code for the system.

5.1 Program Details

The system software consists of seven C files and their associated header files. These programs are `lost2.c`, which is the main system program, the system observer code, which is contained in `velocity.c` and `terrain.c`, and the constrained Kalman filter code, which is contained in `kalman.c`, `nonline.c` and `state.c`. There is an additional program `rotate2.c`, which contains the functions necessary to perform the coordinate rotations required by the system. Each C file is described in more detail in the following subsections.

0990706.051101

5.1.1 The lost2 main program

The `lost2.c` program is the main controlling program for the system. This is the file which must be executed by the user in order to run the system. It opens four files for use. These files are the 1) bathymetric data in UTM triples in meters (depth being negative), 2) the distance traveled by the body since the last sample (i.e. $v * t$) and the measured depth in meters (after pitch, roll and heave are corrected for), and 3) the slant range in meters and the location of the known point in UTM x, y meters. The final file is the output file which will contain the position of the vessel in UTM x, y meters. This file prompts the user for an initial position, desired heading in degrees, and the number of points to be positioned. It then calls each of the functions n times, where n in the number of points to be positioned.

The header file for `lost2.c` contains all the constants in the program which are the tunable parameters in the code. These are `K2` which is the mle adjustment parameter, `MUA` and `MUC` which are the adaptive constants for the along and cross track directions respectively, and `THRES1` which is the along track threshold and `THRES2` which is the cross track threshold. To adjust these parameters, edit `lost2.h` and then use the makefile to recompile the program.

5.1.2 State Observer

The state observer section of the C code in contained in two files. These two files are `velocity.c` and `terrain.c`. The `velocity.c` file corresponds to the state velocity update module and the `terrain.c` file contains the terrain matching module. The detailed internal input/output description is contained later in this chapter.

09905706.051101

velocity.c

The first system module is the state velocity update. The code for this module is contained in the file `velocity.c`. This module is contained in the function velocity. This function takes as input the predicted x,y location of the vessel in UTM and the measured distance traveled since the last sample and outputs the update position prediction. This update is performed by adding the change in x to the x prediction and likewise adding the change in y to the y position.

terrain.c

The main part of the observer is the terrain matching module. The code for this is contained in the file `terrain.c`. `Terrain.c` takes as its inputs the predicted x and y locations of the vessel, the ocean depth measured by the vessel, and a file pointer pointing to the bathymetry. It then computes and outputs the most likely location of the body, given the available inputs, by means of the maximum likelihood function.

5.1.3 Constrained Extended Kalman Filter

The constrained extended Kalman filter is coded in three files. These files are `kalman.c`, `nonline.c` and `state.c`. The `kalman.c` file is the code for a linear extended Kalman filter. The `nonline.c` file contains the code for the nonlinear constraints in the filter. The `state.c` contains the code to predict the next vessel location based on all information available to the current time. The detailed input output follows.

0990706.051101

kalman.c

The first module in the constrained extended Kalman filter is the linear extended Kalman filter. This module takes as input the predicted and observed x and y locations of the vessel, the desired heading, and the initialization point. The heading and initialization point are used throughout the filter segment to rotate between the internal coordinate system and the UTM coordinate system. The innovations are then computed and a portion are added to the prediction to form the filtered vessel position which forms the output.

nonline.c

The next module is the nonlinear constraints module. This function takes as its input the predicted and the filtered position estimates, the slant range and the position of the slant range transponder, and the necessary rotational information. It then thresholds the filtered position and corrects for the slant range, outputting the final system position estimate, in UTM meters.

state.c

The final module in the filter section is the state updater. This module estimates the next position by adding part of the offset between the final and predicted locations to the final position estimate. The inputs are the current prediction and final position estimate both in UTM and outputs an estimate of the next vessel position.

5.1.4 The rotate2 program

The rotate2 file contains two functions that rotate points into and out of the internal coordinate system. The input is the x,y point to be rotated around the

09905706.051101

origin and the desired amount of rotation. After the rotation into the internal coordinate is performed, the along track position is the x axis and the cross track position is the y axis. The rotations into the internal coordinate system are performed by the function rotate2 and rotations back to UTM are performed by the function rotate2b. The output is the point in the other coordinate system.

5.2 The Complete Computer Code

The following subsections contain the code required to run the LOST2 system. The header files are presented first followed by the c files.

5.2.1 The Header Files

The following subsections are the header files for the LOST2 system.

lost2.h

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define K2 10
#define MUA .1
#define MUC .1
#define THRES1 1
#define THRES2 1
```

velocity.h

```
void velocity(double *prex,double *prey,double velox,double veloy);
```

09905706.051101

terrain.h

```
void terrain(double prex,double prey,double depth,double *estx,
double *esty,double x[][3]);
```

kalman.h

```
void kalman(double prex,double prey,double *estx,double *esty,
double heading,double firstx,double firsty);
```

nonline.h

```
void nonline(double prex, double prey,double estx,double esty,
double *finalx,double *finaly,double range,double posx,
double posy,double heading,double firstx,double firsty);
```

state.h

```
void state(double *prex,double *prey,double finalx,double finaly,
double heading,double firstx,double firsty);
```

rotate2.h

```
void rotate2(double *x,double *y,double heading);
void rotate2b(double *x,double *y,double heading);
```

5.2.2 The Program Files

The following subsections are the program files needed for the LOST2 subsystem.

lost2.c

```
#include"lost2.h"
```

```
#include"terrain.h"
#include"velocity.h"
#include"kalman.h"
#include"nonline.h"
#include"state.h"
#include"rotate2.h"


int main(void){

FILE *fp1;    /*the bathy file*/
FILE *fp2; /*the vessel velocity and depth*/
FILE *fp3; /*the slant range and position*/
FILE *fp4; /*the estimated location*/

double firstx;    /*the vessels first x location*/
double firsty;    /*the vessels first y location*/
double estx;    /*the vessels estimated x location*/
double esty;    /*the vessels estimated y location*/
double prex;    /*the vessels predicted x location*/
double prey;    /*the vessels predicted y location*/
double finalx;    /*the vessels final x location*/
double finaly;    /*the vessels final y location*/

int i; /*loop counter*/
int runlength; /* the number of samples to be positioned*/
```

```
double mdepth; /*the meassured depth*/
double range;   /*the meassured slant range*/
double heading; /*the vessels intended heading*/
double posx; /*the slant ranges known x position*/
double posy; /*the slant ranges known y position*/
double velox; /*the x direction velocity*/
double veloy; /*the y direction velocity*/

/*prompt for and input starting position estimate and length of
run, in samples*/

printf("Enter the vessel's estimated starting location, in UTM x\n");
scanf("%lf",&firstx);
printf("Enter the vessel's estimated starting location, in UTM y\n");
scanf("%lf",&firsty);
printf("Enter the vessel's assigned heading\n");
scanf("%lf",&heading);
printf("Enter the number of samples to be positioned\n");
scanf("%d",&runlength);

fp1=fopen("line5a.ascii","r");
fp2=fopen("line5v.nav","r");
fp3=fopen("line5s.dat","r");
fp4=fopen("line5f.nav","w");

/*intialize the system*/
```

```
prex=firstx;
prey=firsty;

for(i=0;i<runlength;i++){
/*get the velocity and the depth*/

fscanf(fp2,"%lf",&velox);
fscanf(fp2,"%lf",&veloy);
fscanf(fp2,"%lf",&mdepth);

/*get the range and position of slant range*/
fscanf(fp3,"%lf",&range);
fscanf(fp3,"%lf",&posx);
fscanf(fp3,"%lf",&posy);

/*system observer*/
/*state velocity update*/
velocity(&prex,&prey,velox,veloy);
/*terrain match module*/
terrain(prex,prey,mdepth,&estx,&esty,fp1);

/*constrained extended kalman filter*/

/*linear kalman filter*/
kalman(prex,prey,&estx,&esty,heading,firstx,firsty);
```

```
/*nonlinear constraints*/

nonline(prex,prey,estx,esty,&finalx,&finaly,range,posx,posy,heading,
firstx,firsty);

/*state prediction*/
 state(&prex,&prey,finalx,finaly,heading,firstx,firsty);

/*output the estimated location*/

fprintf(fp4, "%f %f\n",finalx,finaly);
}

return(0);
}
```

velocity.c

```
#include"lost2.h"
#include"velocity.h"

void velocity(double *prex,double *prey,double velox,double veloy){

/*as of right now the velocity is assumed to be meters per
sample*/
```

```
/*project the position ahead*/
*prex=*prex+velox;
*prey=*prey+veloy;

return;
}
```

terrain.c

```
#include"lost2.h"
#include"terrain.h"
void terrain(double prex,double prey,double depth,double *estx,
double *esty,FILE *fp1){

double minerror;      /*min error found*/
double error;          /*probability that you're at that point*/
double xmin, ymin; /*most likely point*/
double x,y,z; /*The bathy triple*/

rewind(fp1);

/*intialize the search*/
minerror=100000000000;
xmin=0;
ymin=0;
```

```
/*search the enitre bathy set for the point that minimizes the
mle*/

fscanf(fp1,"%lf",&x);
while(x!=EOF){
fscanf(fp1,"%lf %lf",&y,&z);
error = pow((x-prex),2) + pow((y-prey),2);
error = error+K2*pow((z-depth),2);
if(error<minerror){
minerror=error;
xmin=x;
ymin=y;
}
fscanf(fp1,"%lf",&x);
}

/*report that value to main*/
*estx=xmin;
*esty=ymin;


return;
}
```

kalman.c

```
#include"lost2.h"
#include"kalman.h"
#include"rotate2.h"


void kalman(double prex,double prey,double *estx,double *esty,
double heading,double firstx,double firsty){

double alongp; /*the along track position prediciton*/
double crossp; /*the cross track position prediciton*/
double alonge; /*the along track position  estimate*/
double crosse; /*the cross track position estimate*/
double innova; /*the along track inovations at the
current time*/
double innovc; /*the cross track inovations at the
current time*/


/*rotate into the along track cross track coordinate system*/

alongp=prex-firstx;
crossp=prey-firsty;
alonge=*estx-firstx;
crosse=*esty-firsty;

/*peform the rotation*/
```

```
rotate2(&alongp,&crossp,heading);
rotate2(&alonge,&crosse,heading);

/*find the innovations*/

innova=alonge-alongp;
innovc=crosse-crossp;

/*preform the linear Kalman filtering*/

*estx=alongp+MUA*(innova);
*esty=crossp+MUA*(innovc);

return;
}
```

nonline.c

```
#include"lost2.h"
#include"nonline.h"
#include"rotate2.h"

void
nonline(double prex, double prey,double estx,double esty,
double *finalx,double *finaly,double range,double posx,
double posy,double heading,double firstx,double firsty){
```

```
double rprex; /*rotated position x est*/
double rprey; /*rotated position y est*/
double innova; /*the along track innovations*/
double innovc;  /*the cross track innovations*/
double scale;   /*the scale factor to correct for the slant
range*/
double distx;   /*x distance to the transponder*/
double disty; /*y distance to the transponder*/


/*compute the rotated position*/
rprex=prex-firstx;
rprey=prey-firsty;

/*peform the rotation*/
rotate2(&rprex,&rprey,heading);

/*find the innovations*/

innova=estx-rprex;
innovc=esty-rprey;

/*apply the constraints*/

if(innova<THRES1){
```

```
if(innova>-THRES1){
*finalx=estx;
}
else{
*finalx=rprex-THRES1;
}
}
else{
*finalx=rprex+THRES1;
}
if(innovc<THRES2){
if(innovc>-THRES2){
*finaly=esty;
}

else{
*finaly=rprey-THRES1;
}
}
else{
*finaly=rprey+THRES1;
}

/*convert to real world corridinates*/

rotate2b(finalx,finaly,heading);
```

```
*finalx=*finalx+firstx;
*finaly=*finaly+firsty;

/*apply the slant range*/
distx=*finalx-posx;
disty=*finaly-posy;
scale=range/sqrt((pow(distx,2)+pow(disty,2)));
distx=distx*scale;
disty=disty*scale;

/*compute the position*/

*finalx=posx+distx;
*finaly=posy+disty;

return;
}
```

state.c

```
#include"lost2.h"
#include"state.h"
#include"rotate2.h"

void state(double *prex,double *prey,double finalx,double finaly,
double heading,double firstx,double firsty){
```

```
double alongp; /*the along track position prediciton*/
double crossp; /*the cross track position prediciton*/
double alonge; /*the along track position  estimate*/
double crosse; /*the cross track position estimate*/
double innova; /*the along track inovations at the
current time*/
double innovc; /*the cross track inovations at the
current time*/


/*rotate into the along track cross track coordinate system*/

alongp=*prex-firstx;
crossp=*prey-firsty;
alonge=finalx-firstx;
crosse=finaly-firsty;

/*peform the rotation*/
rotate2(&alongp,&crossp,heading);
rotate2(&alonge,&crosse,heading);

/*find the innovations*/

innova=alonge-alongp;
innovc=crosse-crossp;
```

```
/*preform the forward prediction*/

*prex=alonge+(1-MUA)*(innova);
*prey=crosse+(1-MUC)*(innovc);

rotate2b(prex,prey,heading);

*prex=*prex+firstx;
*prey=*prey+firsty;

return;
}
```

rotate2.c

```
/*Performs 2d rotations about the z axis

input: x,y location and the amount of rotation
output rotated x,y

rev: 1.0

*/
#include "lost2.h"
#include "rotate2.h"
```

```
void rotate2(double *x,double *y,double heading){

double rotamount;    /*ammount to rotate*/
double xold,yold; /*temps to hold orginal x and y*/

/*transform into angles from heading*/

heading=-heading+90;
if(heading<0)
heading=heading+360;

rotamount=heading;


xold=*x;
yold=*y;
*x=xold*cos(rotamount)-yold*sin(rotamount);
*y=xold*sin(rotamount)+yold*cos(rotamount);
return;}

void rotate2b(double *x,double *y,double heading){

double rotamount;    /*ammount to rotate*/
double xold,yold; /*temps to hold orginal x and y*/
```

```
/*transform into angles from heading*/

heading=-heading+90;
if(heading<0)
heading=heading+360;

rotamount=-heading;


xold=*x;
yold=*y;
*x=xold*cos(rotamount)-yold*sin(rotamount);
*y=xold*sin(rotamount)+yold*cos(rotamount);
return;}
```

0990706 .051101

Chapter 6

Simulator Results

In order to determine how the system would perform before actual sea trails, a simulator was designed to produce results similar to would be expected from the real system. These include all the sensors present in the system and the returns of all the sonars. This was accomplished using real bathymetric data collected during a survey and artificially generating the returns of the sensors.

6.1 Data Generation

A computer program was created that takes data points from a previously mapped region and returns files that would be similar to the ones expected from an actual survey system, and are in the format required by the LOST2 system. The simulator was created using a modular concept. The final design generates two data files, to simulate the output of all the sensors used in the system. The files contain the slant range and location of the known point, and the vessels navigation system.

The generated vessel information file has two distinct versions. Because there is no way to know the vessel's real position by means of a sensor, the file generated by the simulator to mimic the sensors cannot contain this information.

However, for any experimental results to be verified, the actual location must be known. The simulator must generate and record the vessel's location but it must be hidden from the output file. Therefore, the simulator was designed to make two data files, one with and one without the vessel's location. The vessel's location was generated by integrating the vessel's velocity and adding a random walk to both the along and the cross track position of the vessel. This has the effect of allowing movement that is expected due to currents. The vessel depth is assumed to be constant because underwater vessel's are often used in constant depth mode.

The second set of measurements taken from the simulator is the ranging device's measurement. The ranging device is a transponder, which provides the range to the vessel from the known location. The simulation of this is accomplished by using the distance formula to calculate the range based on the actual locations of the known position and the vessel. This information composes the last data stream from the simulator.

This simulator provides the capability to generate as many sample runs as desired over any terrain that has been mapped. Due to the manner in which the simulator handles data, there is no dependency on any preprocessing that is performed. This allows for testing over ungridded or gridded sets, where discontinuities have been removed from irregularly, sparsely sampled data sets. By allowing for this flexibility, we are able to test our system over any terrain and sampling conditions for which data exists.

Feeding the data files into the LOST2 system, the output position can be compared to the actual location of the tow fish that was recorded during data generation. This provides a method to quantify the error in the position generated by the LOST2 system. The advantage to computer simulation for testing purposes

0990706.051101

is that we know the location of the vessel at all times. In the real data set, the error in location is not known absolutely, because the real position is taken from some other system which typically will have an error of one percent of range or higher.

6.2 Test Set

For initial evaluation, data collected during a survey of the Pensacola Bay Channel was selected as the test data set. This data set is in the form of ungridded irregularly spaced (x, y, z) triplets. This data set provided a set of various terrains enabling conclusions to be drawn about the system performance under various conditions. The data set is shown in Figure 6.1. North for this data set is straight up and the depth ranges from 8 m to 16 m. The shading in the figure represents the filled depth contours on the bottom.

This data set was chosen for initial testing due to the large number of terrain variations in the set. The contours present have a definite order and direction. The organization of the contours allows for the behavior of the system in relation to the direction of travel to be determined in relation to the direction of the contours.

6.3 Results

The simulator created a method to perform tests on the system. The set of tests involved running the system over various terrains to determine the terrain based performance of the system. Based on performance, the terrains tested could generally be classified as one of three types. In the best terrain area, the prevailing contours run parallel to the track of the survey and are close together.

D990706 .051101

If the contours remain close, but they are perpendicular to the direction of travel, performance falls into the middle performance category. The worst performance occurs over terrain where there are few to no contours present. The terrain based performance section of this chapter details sample results for each performance condition.

Trackline plots and position error plots have been provided for each test. The trackline plots each contain three lines. The dotted line is the estimated vessel track, before any acoustic position correction, is the best estimate of the vessel's location. The dashed line is the real track of the vessel generated by the simulator. The solid line is the LOST2 system's estimated track for the vessel. Following each track is a plot of the difference between the LOST2's estimate and the real course. This error is shown in three parts: total error magnitude; the cross track component of the error; and the along track component of the error. The simulations presented is this section were taken while the vessel was approaching the slant range transponder. This allows for the effects of the slant range to be seen in the along track component of the error. These runs also the same runs used for the LoST system tests[56]. The improvement of LOST2 over LoST is provided as well.

The terrains tested in this set of simulations fall into three performance categories. The best terrain condition is characterized by a rough bottom that tends to have contours that extend parallel to the direction of travel. A sample run over this terrain is shown in Figure 6.2. Due to the zoom in of the track the contours in Figure 6.2 are deceiving. Figure 6.3 provides a breakdown of the error for this condition. As Figure 6.3 shows, most of the error is contained in the cross track estimation. This is expected because the ranging module provides an excellent estimate of the along track location. The terrain match works well

09905706 .051101

under these condition and provides the best cross track estimate of location. The rms error value for this run was 0.54 meters, using the LoST system, and 0.52 meters for the LOST2 system, an improvement of 4 percent. In this area, the predicted position accuracy, as determined by the spatial performance prediction module was 1.27 meters.

The next best terrain for this system is a rough bottom, where the track runs perpendicular to the contours. Due to the fact that the bottom does not change rapidly over the search area, this condition hinders the terrain match. A sample run under this condition is shown in Figure 6.4. The system tends to over compensate for the cross track velocity in this condition causing the estimated course to oscillate around the correct course. This oscillation can be seen in the cross track error part of the error plot contained in Figure 6.5. The rms error values of this run was 0.79 meters with the LoST system and 0.77 meters with the LOST2 system, an improvement of 3 percent. The predicted position accuracy, as determined by the spatial performance prediction module, in this area was 1.28 meters.

The LoST system's worst rms error value of 1.1 meters with is encountered over nearly flat terrain, with a variation of less than three meters. The LOST2 system reduced this error to 1.05 meters, an increase in performance of 5 percent. A representative run is contained in Figure 6.6. The increase in the error in this case is contained in an offset in the cross track component of the error. This offset is shown as Figure 6.7.

6.4 Simulation Conclusions

There were several conclusions from the simulator studies. First, given perfect sensors the system is capable of locating the vessel to within the resolution of the

0990706.051101

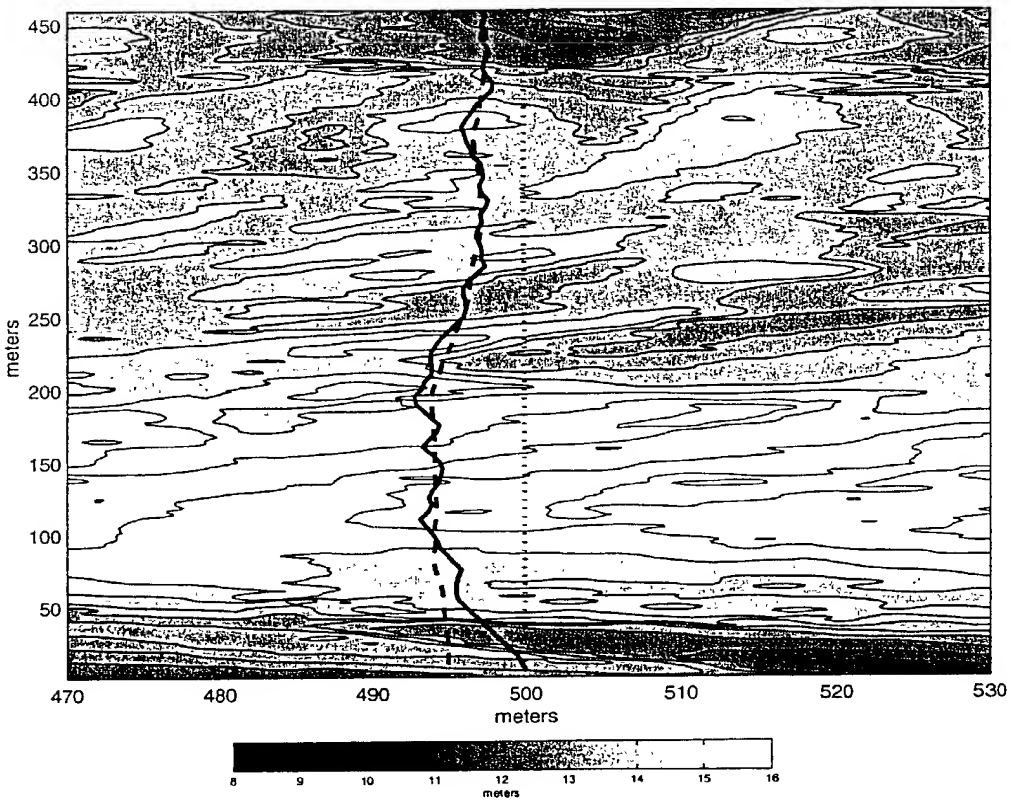

Figure 6.2: Best Operating Conditions
Solid Line - Estimated Course
Dashed Line - Real Course
Dotted Line - Best Estimate Before LoST System

0990706.051101

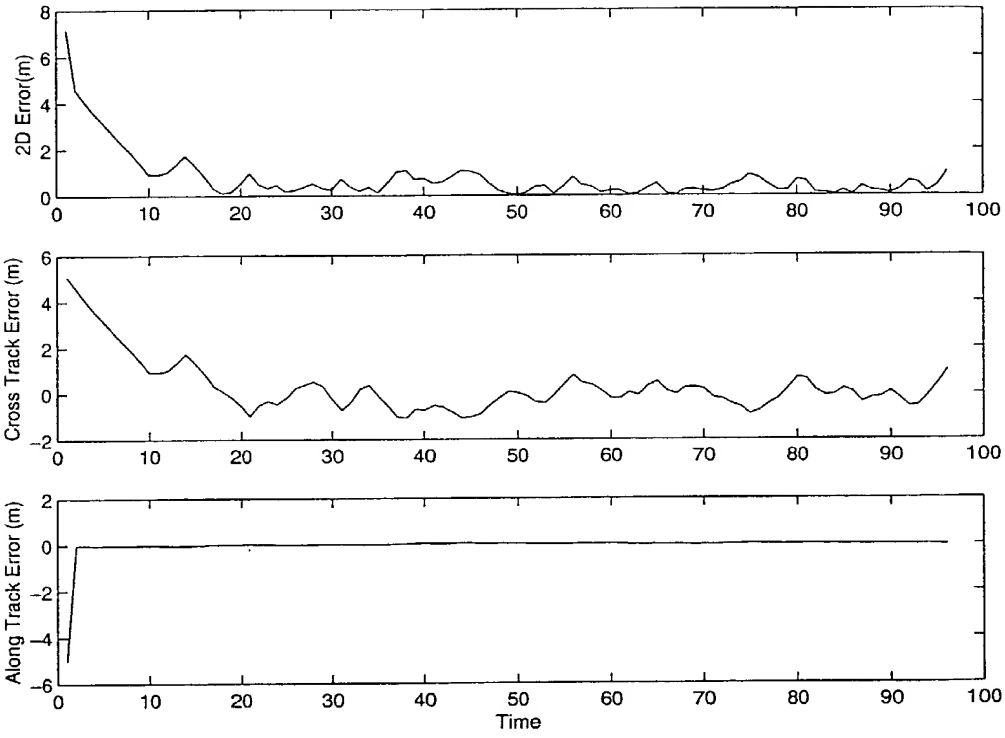

Figure 6.3: Error Plots for Best Operating Conditions

D9905706.051101

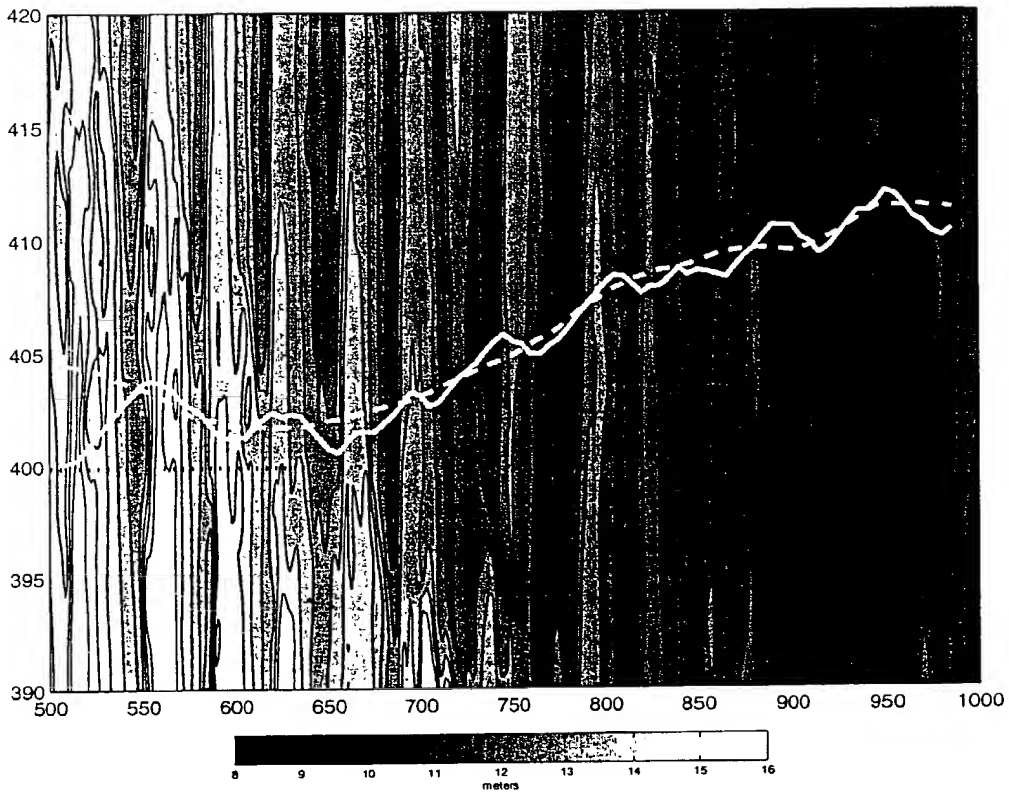

Figure 6.4: Midcase Operating Conditions
Solid Line - Estimated Course
Dashed Line - Real Course
Dotted Line - Best Estimate Before LoST System

0990706 .051101

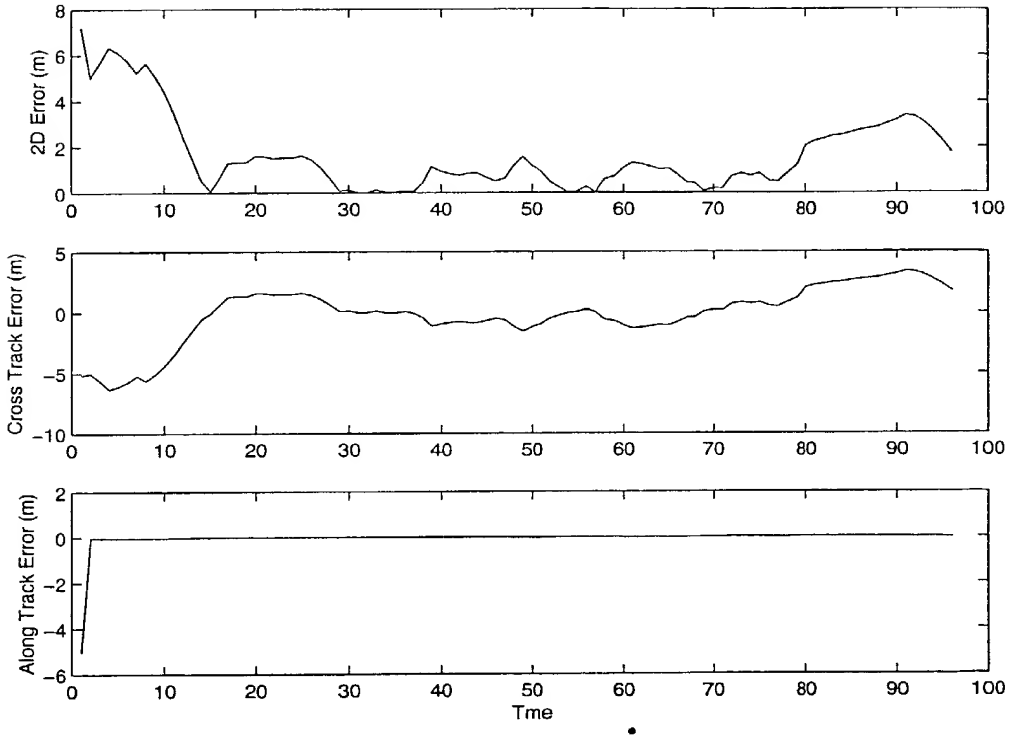

Figure 6.5: Error Plots for Midcase Operating Conditions

09905706 .051101

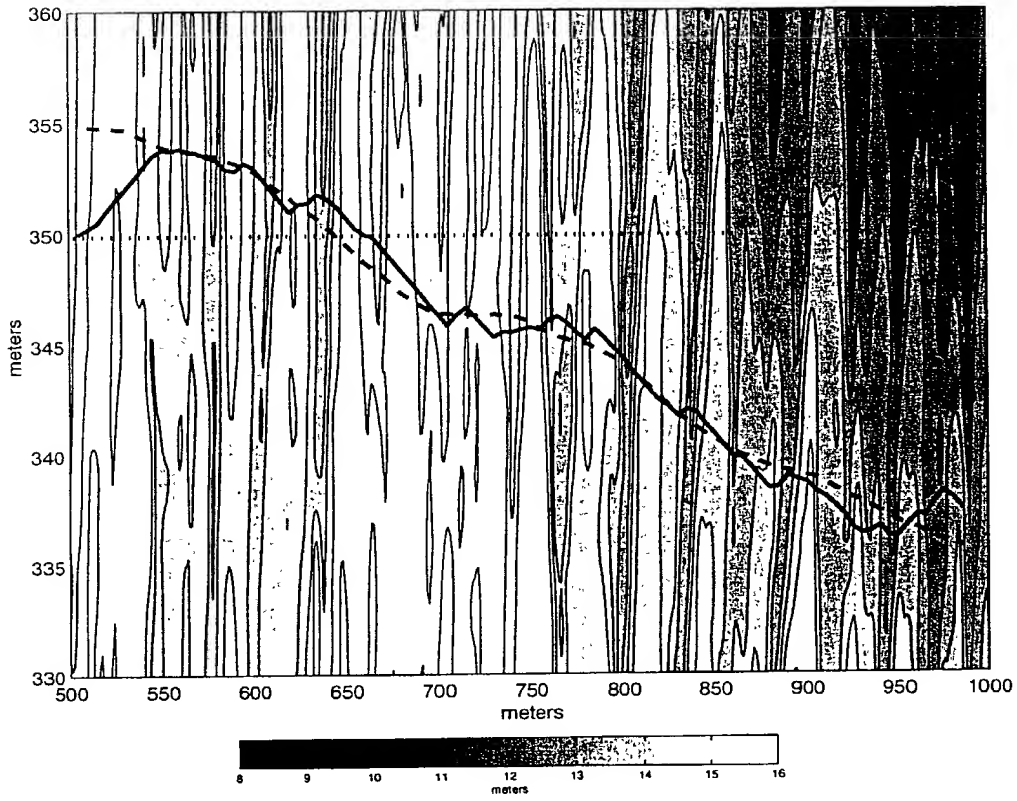

Figure 6.6: Marginal Operating Conditions
Solid Line - Estimated Course
Dashed Line - Real Course
Dotted Line - Best Estimate Before LoST System

09905706.051101

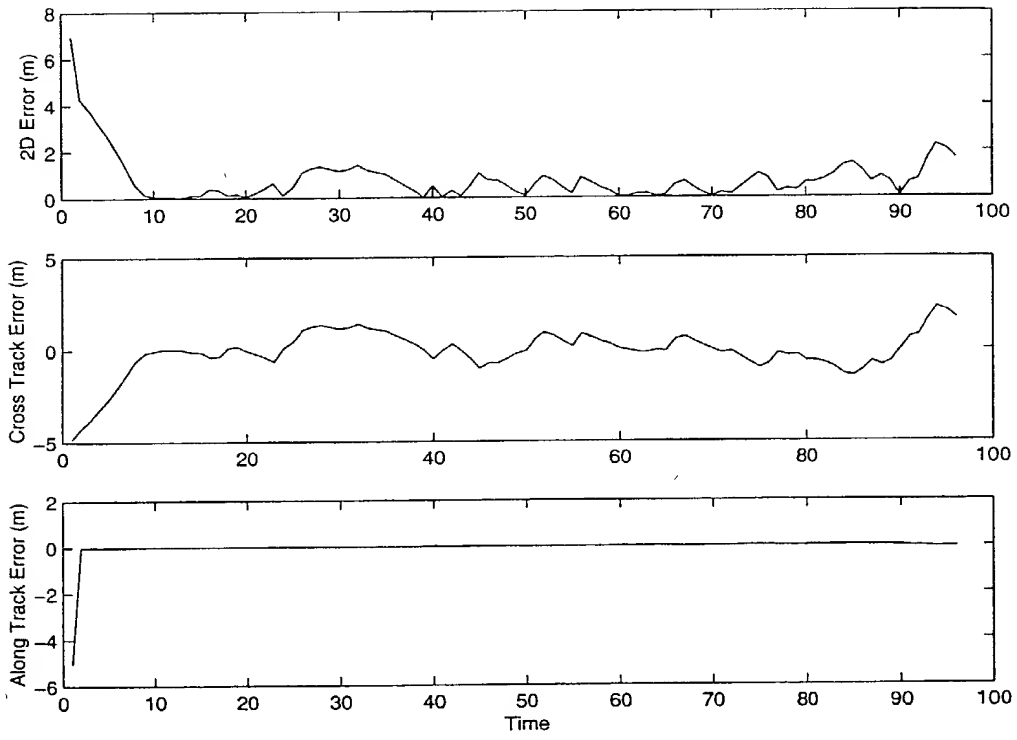

Figure 6.7: Error Plots for Marginal Operating Conditions

0990706.051101

bathymetry. Second, The terrain match performance is directly related to the amount of variation on the bottom. The final conclusion was that the system performance in simulation was sufficient to lead to the decision to get real data from a real mission with an independent positioning system operating, in order to verify these results using real sensors. This data will in conjunction with the simulator results, provide proof of concept of the system. The real data is described in detail in the next chapter.

0990706.051101

Chapter 7

Real World Data Documentation

This chapter presents the towfish data that has been provided by C&C Technologies, Inc. for validation of the monument based positioning system. This data was collected in the Gulf of Mexico from 12 Jan. to 20 Jan. 1999. This was a two ship towbody operation. The Ocean Surveyor towed the towbody and collected the bathymetry and the David McCall provided USBL positioning for the towbody. The top level directory for this data is `towfishcc`. This directory contains two subdirectories, `fishdata` and `rawbathy`. The `fishdata` directory contains the data from the towfish, while the `rawbathy` directory contains the raw ungridded bathymetry from the survey area.

7.1 The Fishdata Directory

This directory contains the data collected from the towfish. This data is from lines labeled 5, 6 and 7. The files are named `LINE5.txt`, `LINE6.txt` and `LINE7.txt`. Each of these files contains approximately 44,000 samples, covering about three hours of the ship's state (position and heading), the tow's state (three dimensional position, pitch, roll, heading and speed), and the water depth all referenced to a common time reference.

Internally each file is organized into thirteen columns. These columns correspond to time, ship latitude, ship longitude, ship heading, fish north, fish east, fish speed, fish heading, fish roll, fish pitch, fish depth, fish altitude and water depth. Each of these is explained in the following short paragraphs. Plots from the `LINE5.txt` file are included as appropriate to illustrate the data. The first three lines of this file are included here.

15:41:18.400 2755.326900 9041.403400 160.9 10139579.0 2386221.2 9.8 132.5 3.30 8.40 413.5 28.0 441.56

15:41:18.400 2755.326900 9041.403400 160.9 10139579.0 2386221.2 9.8 132.5 3.30 8.40 413.5 28.0 441.56

15:41:18.400 2755.326900 9041.403400 160.9 10139579.0 2386221.2 9.8 132.5 3.30 8.40 413.5 28.0 441.56

The first column is time. The time is represented as a 24 clock in the form of HH:MM:SS.SSS. The time zone that this time is referenced to is CST. It is unknown at this point where the time sensor was mounted. There is no figure for this column.

The second column in the data is the ship latitude. This data is plotted vs. sample number in figure 7.1. A zoomed in version is contained in figure 7.2. The unit of this number is DDMM.MMM where D is degrees and M is minutes.

The third column contains the ship's longitude. This data is plotted vs. sample in figure 7.3. A zoomed in region is shown in figure 7.4. Again the unit of this column is DDMM.MMM.

The second and third columns represent the track of a ship. This track is the track of the USBL ship, the David McCall. These two columns are plotted against each other in figure 7.5. A zoomed in section showing the sample to sample relationship is contained in figure 7.6. There is a noticeable chatter around a mean

09905706 .051101

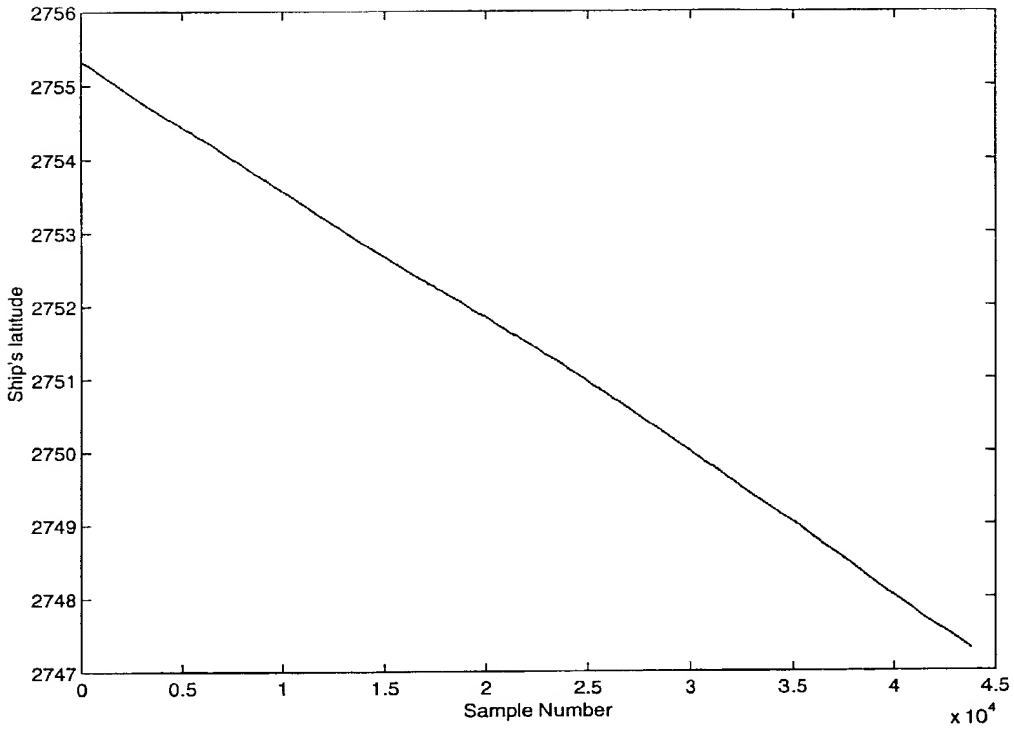

Figure 7.1: The ship's latitude plotted vs. sample number.

09905706.051101

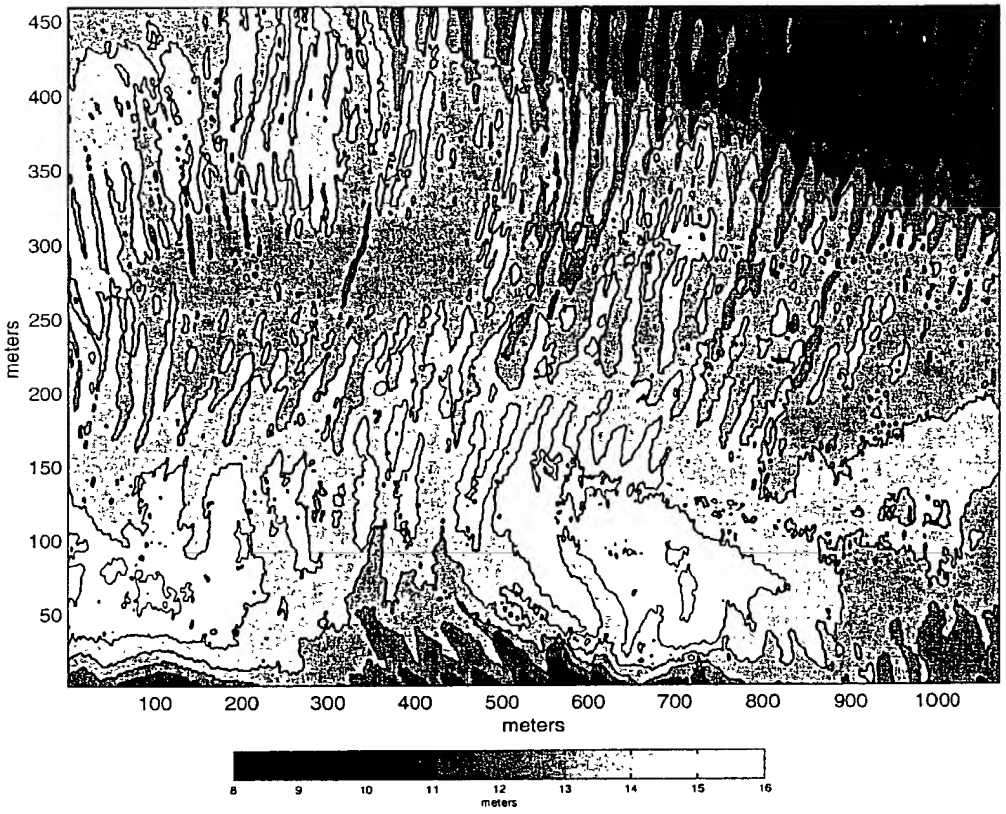

Figure 6.1: Pensacola Bay Data Set used in Testing and Evaluation

09905706.051101

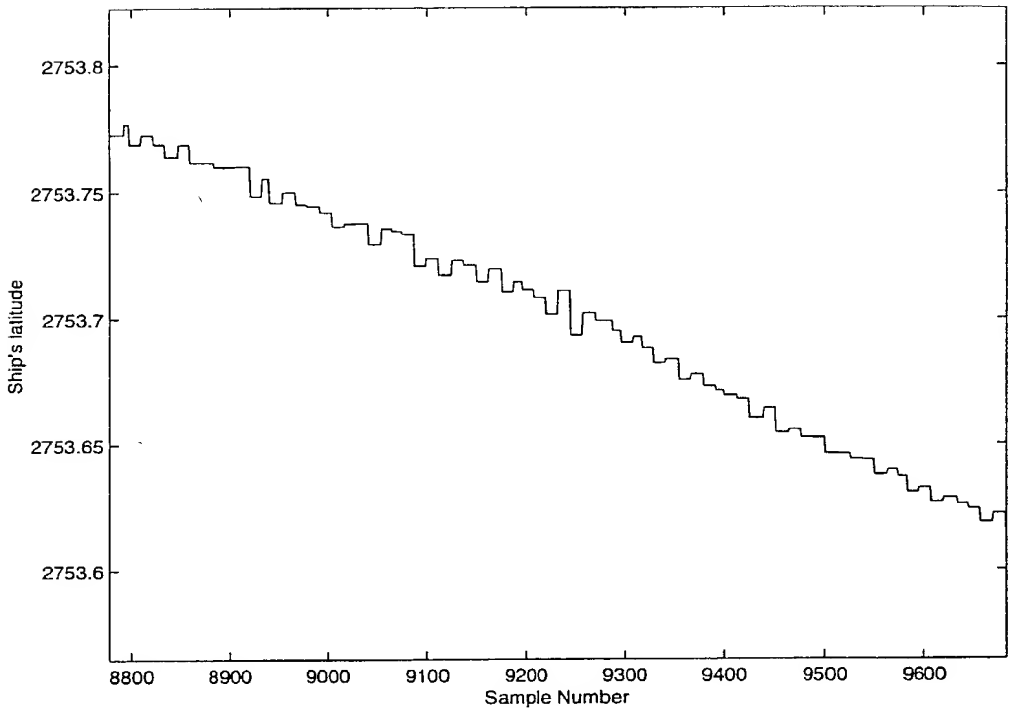

Figure 7.2: The ship's latitude plotted vs. sample number.

0990570606.051101

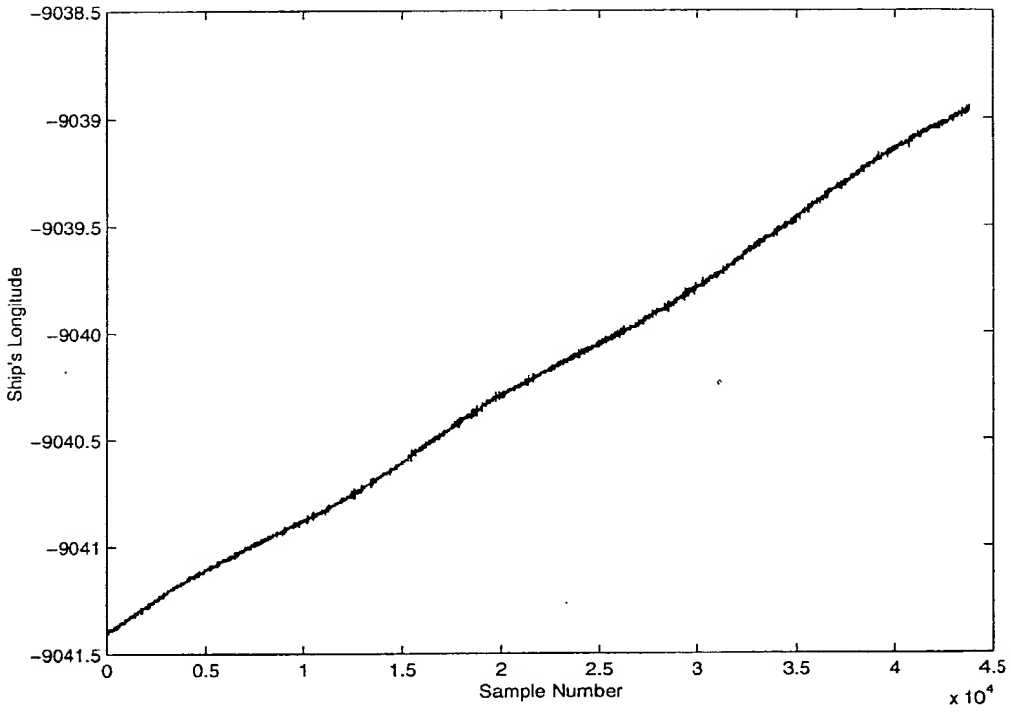

Figure 7.3: The ship's longitude plotted vs. sample number.

0990706.051101

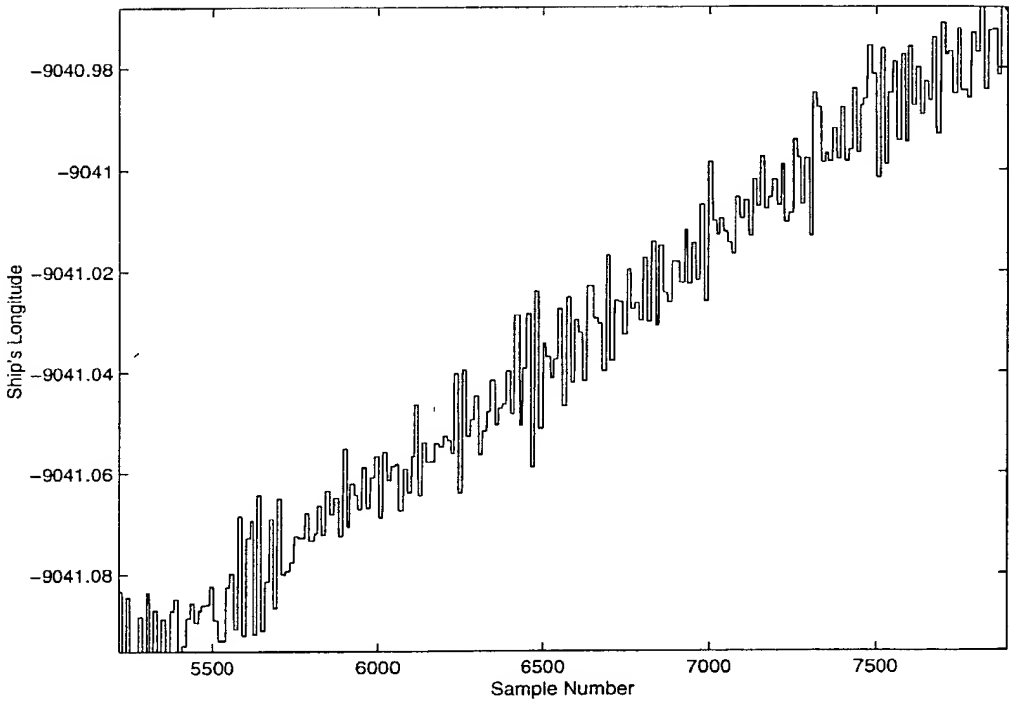

Figure 7.4: The ship's longitude plotted vs. sample number.

path in the ship's track. The source of this is unknown. Also, this track appears to be rotated 90 degrees from the towfish track. However, this was in the western hemisphere and the larger numbers of longitude mean the position is further west. Therefore, these values should be plotted as negative numbers but that is not how they are stored.

The forth column is the ship heading. As is readily evident is figure 7.7, there was a problem with the heading sensor. The heading does not correspond to the ship track, nor is it a realizable ship heading. The unit on heading is degrees.

The fifth column represents the towfish's north position. This position is plotted vs. sample number in figure 7.8. A zoomed version is contained in figure 7.9. This position is in feet, referenced to the WGS-84 datum. If converted to meters, the UTM grid this position is 15 North.

The sixth column represents the towfish's east position. This position is plotted vs. sample number in figure 7.10. The intersample relationship is shown in figure 7.11. This position is in feet on the WGS-85 datum. Again if converted to meters, this position is referenced to UTM 15 north.

These two columns define the tow's track. The plot of the tow's line5 track in feet is contained in figure 7.12. A smaller segment is shown in more detail in figure 7.13. This corresponds to the vessel track.

The seventh column is the tow's speed. This speed is plotted vs. sample in figure 7.14. The unit on the speed is knots and what it is measured with respect to is unknown. Careful attention was not paid to this data, probably because it was not needed for C&C's purposes.

The eighth column of the data file is tow heading. The heading is plotted vs. sample in figure 7.15. This represents a compass heading in degrees. The numbers agree with the tow track, and so this data appears to be reliable.

0990706.051101

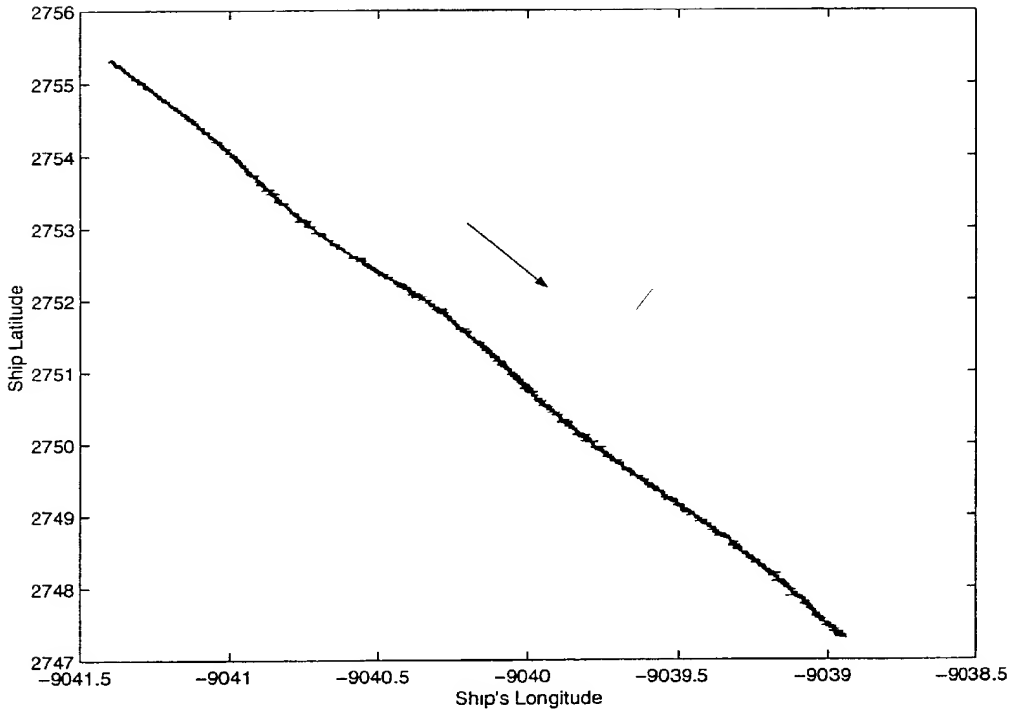

Figure 7.5: The ship's longitude plotted vs. ship's latitude.

09905706.051101

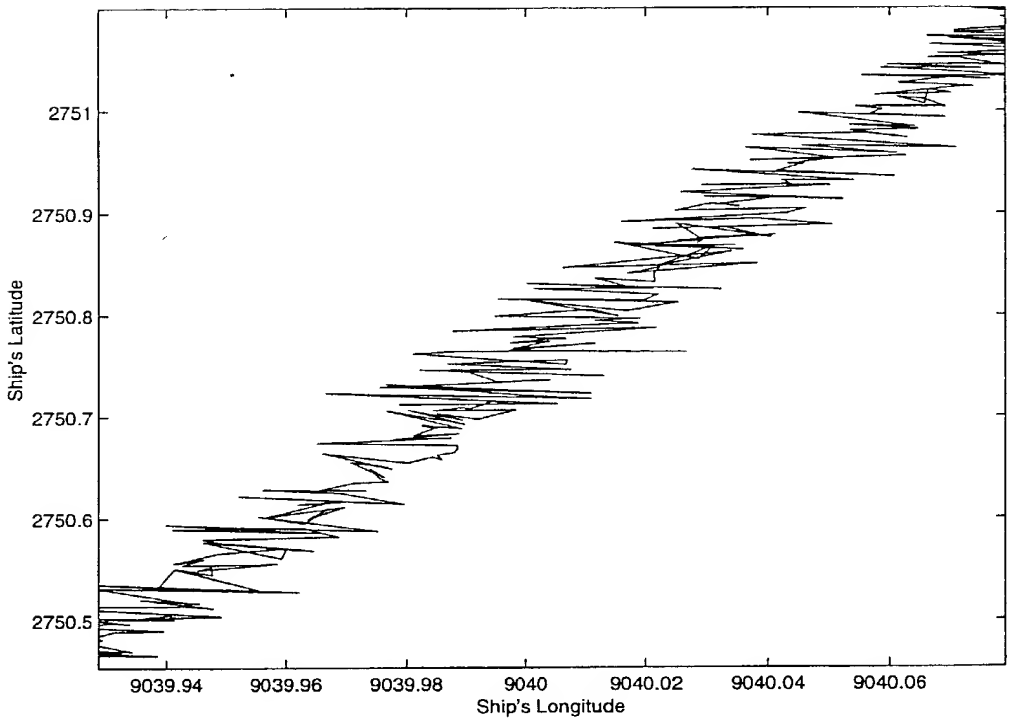

Figure 7.6: The ship's longitude plotted vs. ship's latitude.

09905706.051101

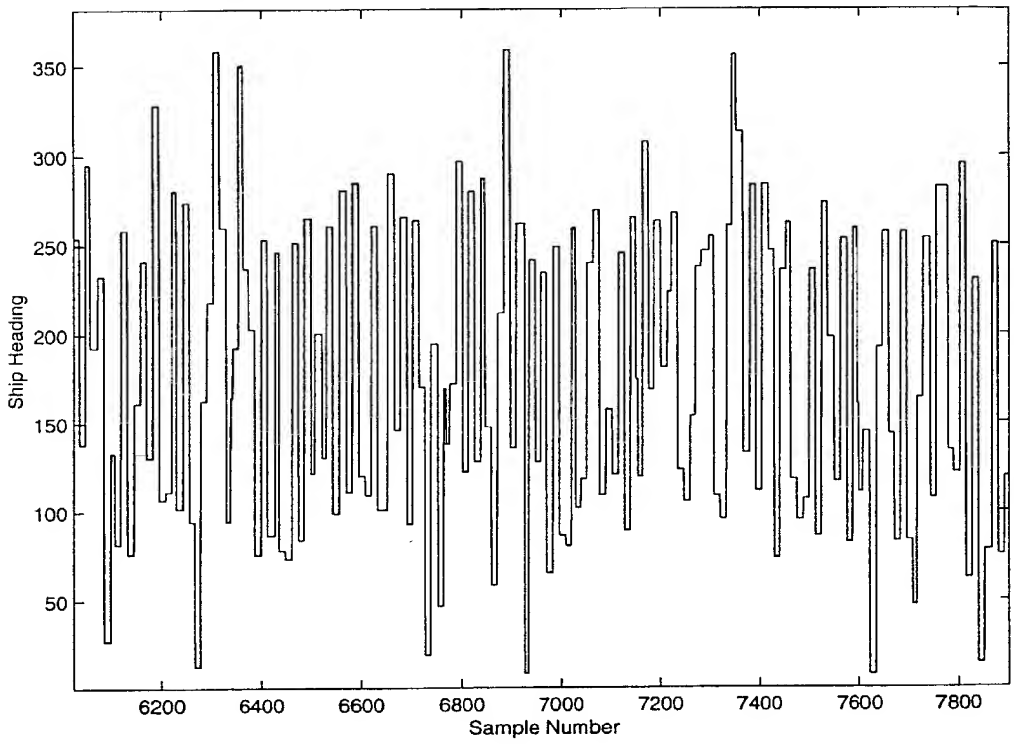

Figure 7.7: The ship heading plotted vs. sample number.

09905706.051101

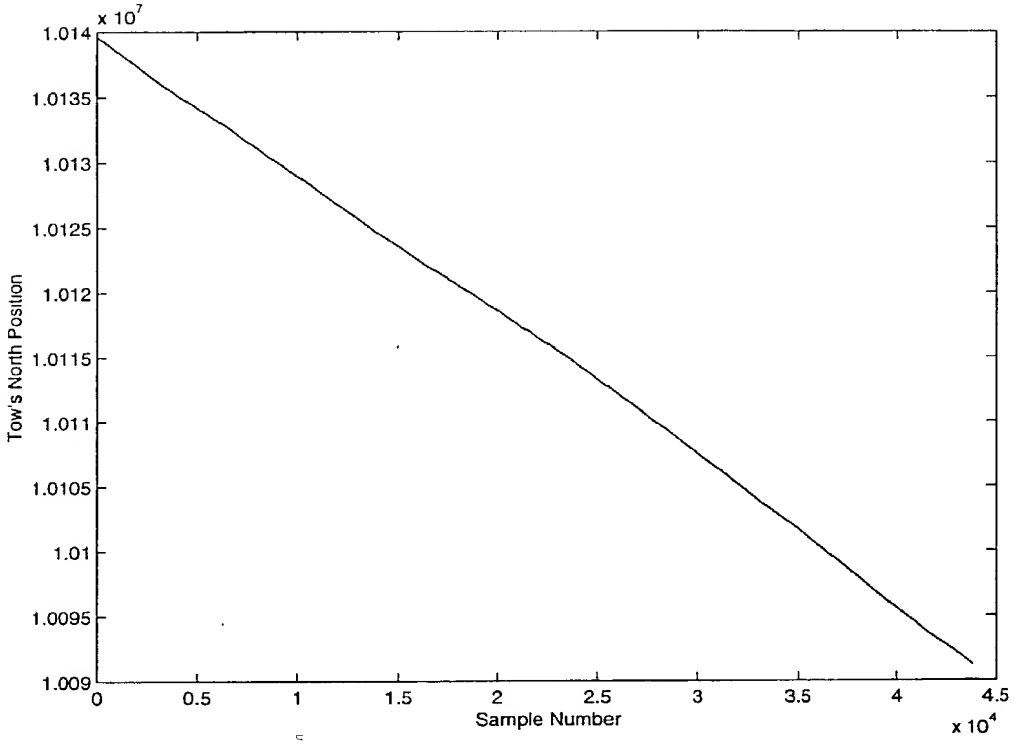

Figure 7.8: The tow's north position plotted vs. sample number.

09905706 .051101

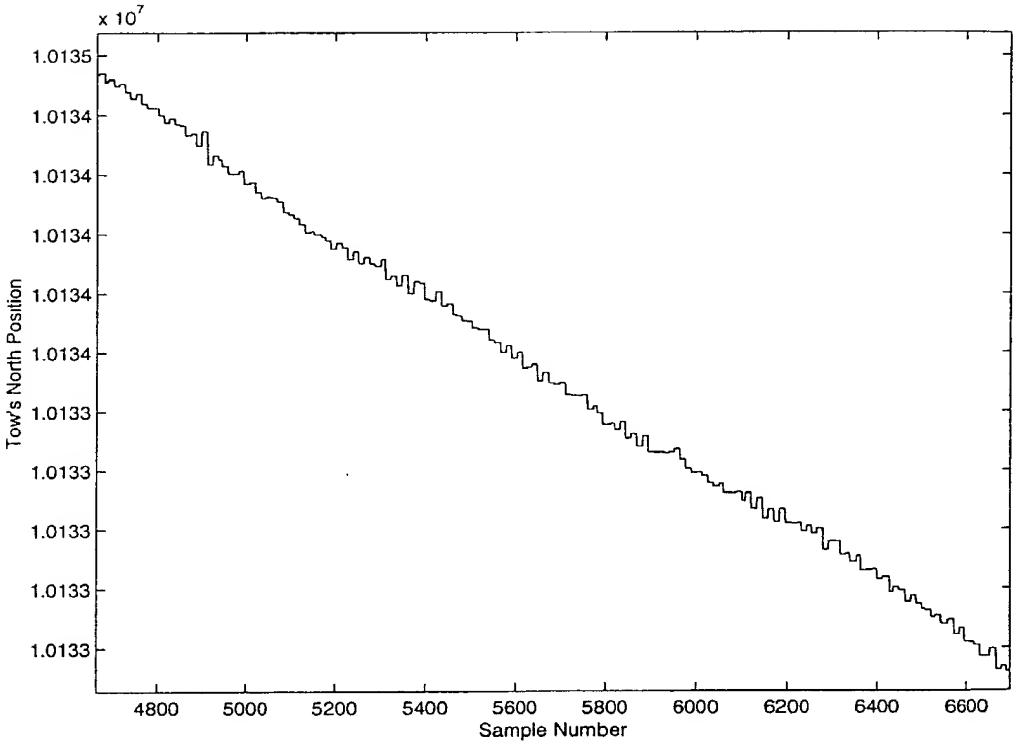

Figure 7.9: The tow's north position plotted vs. sample number.

09905706.051101

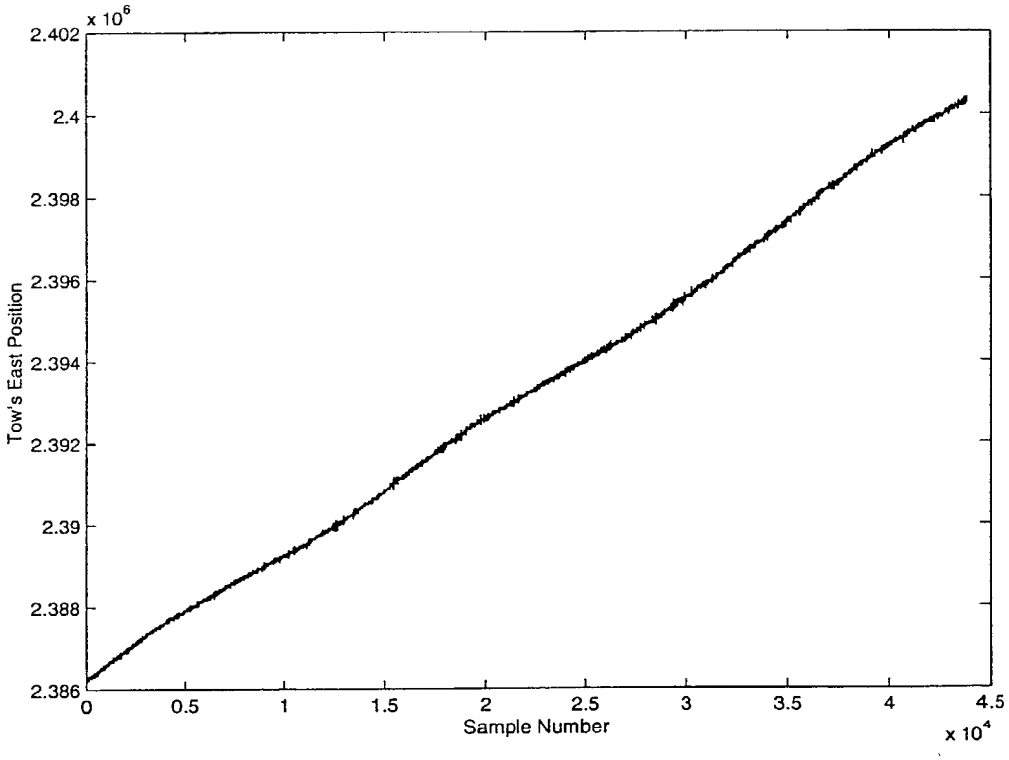

Figure 7.10: The tow's east position plotted vs. sample number.

0990506.051101

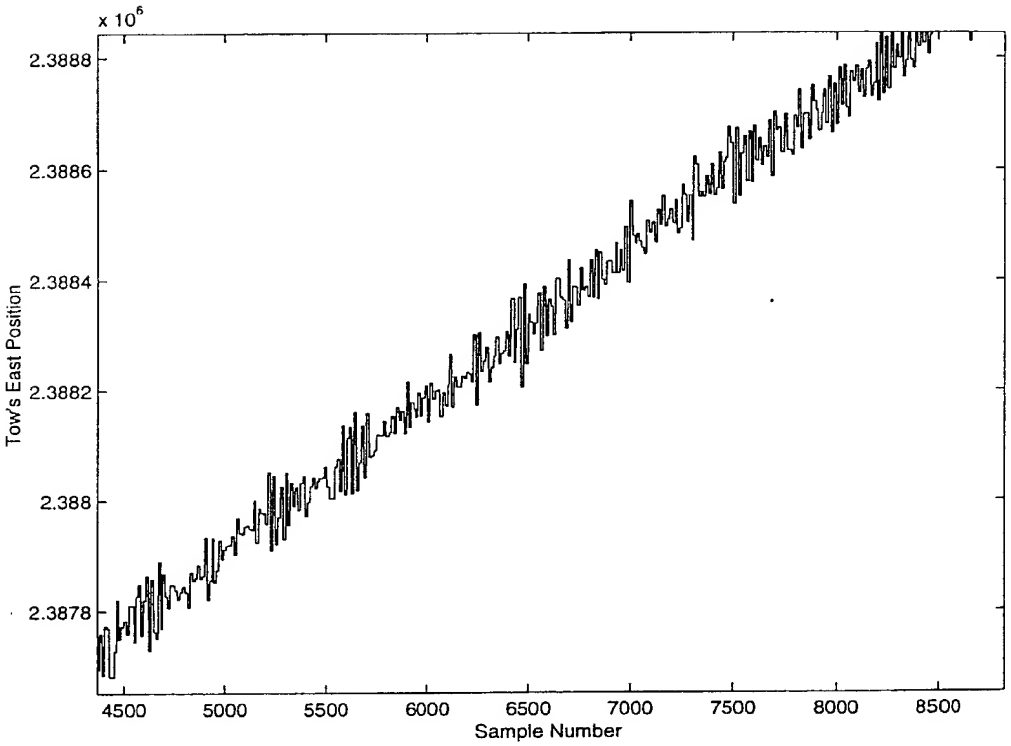

Figure 7.11: The tow's east position plotted vs. sample number.

0990706.051101

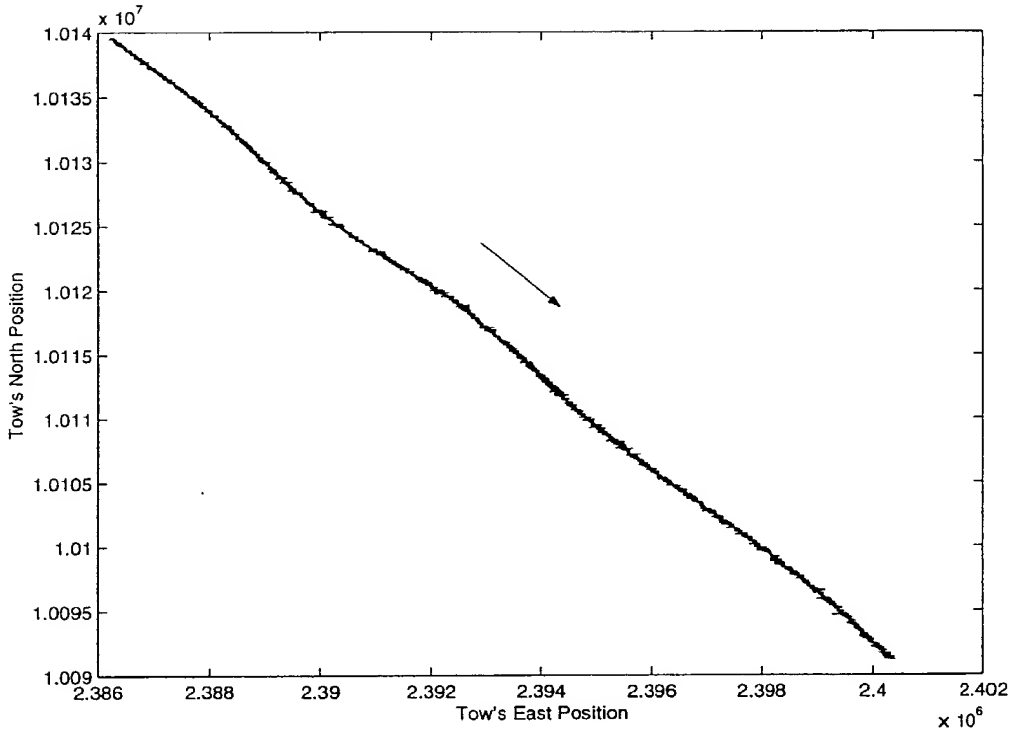

Figure 7.12: The tow's track, in UTM region 15 North.

0990570s .051101

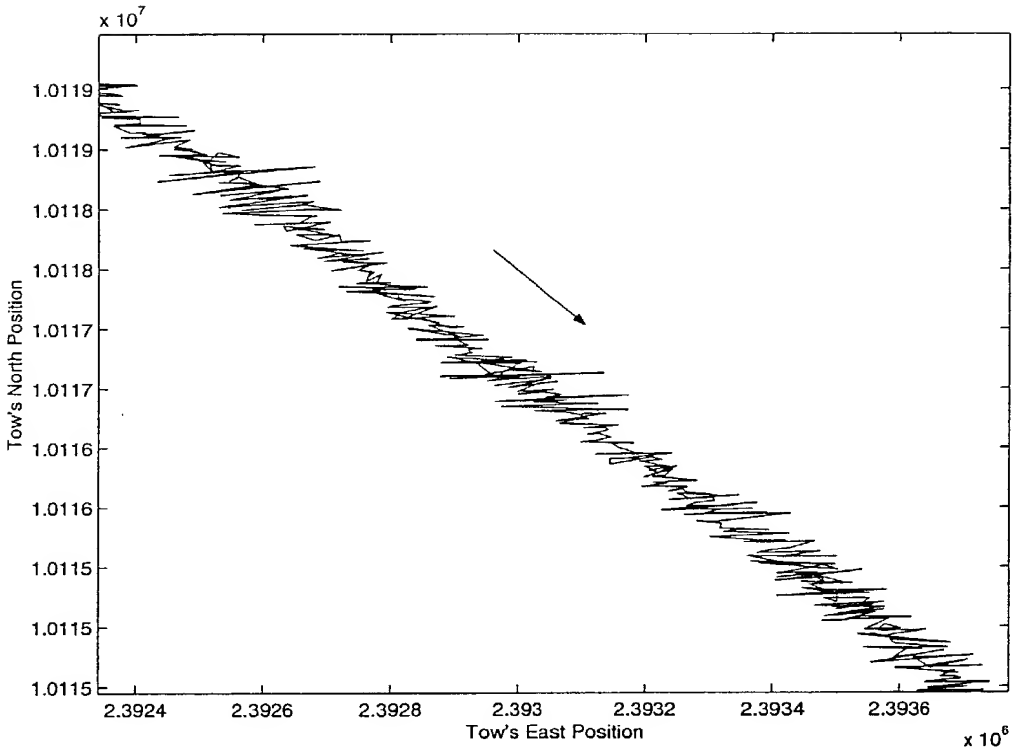

Figure 7.13: The tow's track, in UTM region 15 North.

0990706.051101

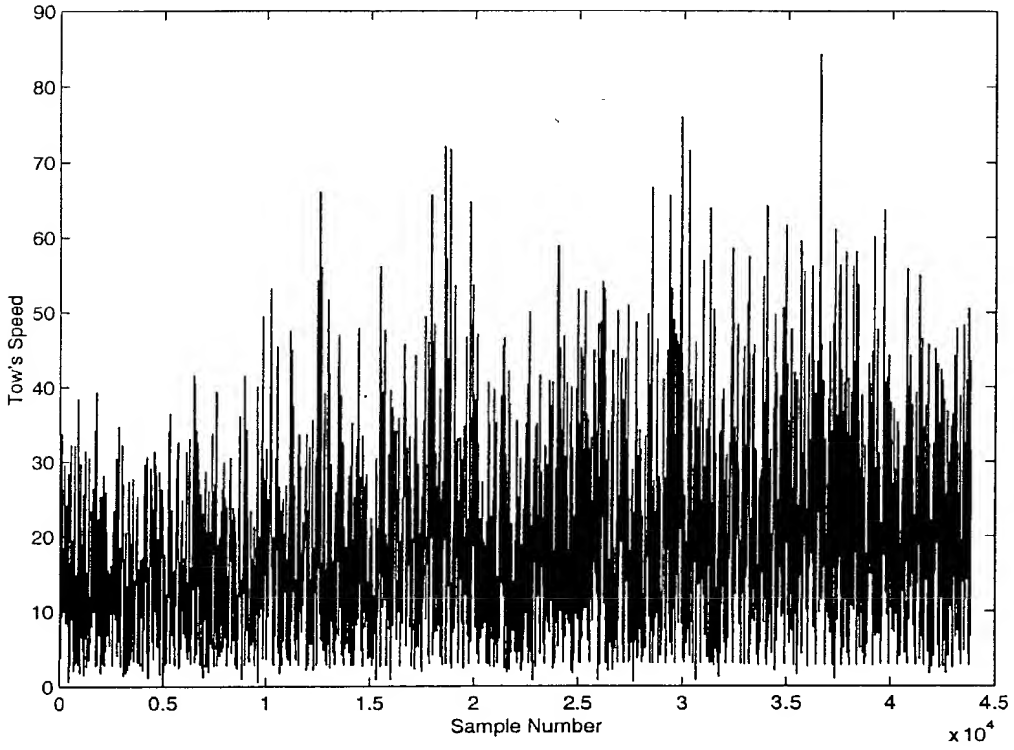

Figure 7.14: The tow's speed, in knots with respect to the water column.

0990570606.051101

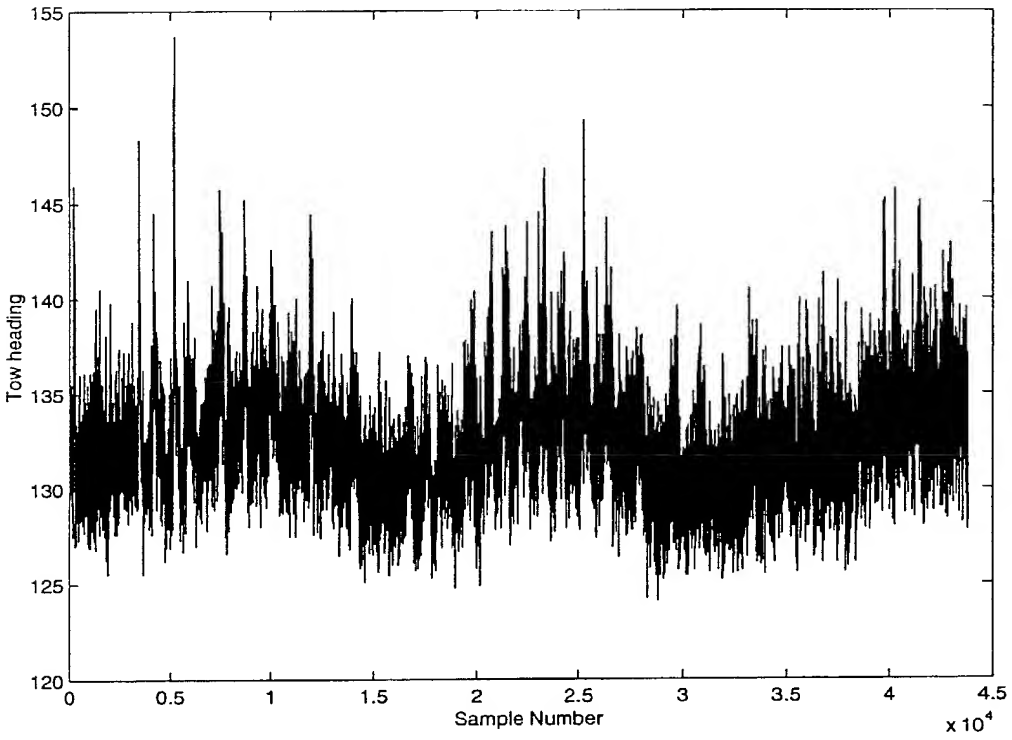

Figure 7.15: The tow's heading over line 5.

The ninth column is the towfish's roll. The roll plotted vs. sample is shown as figure 7.16. The roll over a shorter period of time is shown in figure 7.17. This appears to be reasonable and measured in degrees. The sense of positive roll angles is positive roll, starboard down.

The tenth column is the towfish's pitch. The tow's pitch is shown in figure 7.18. A zoomed version is provided in figure 7.19. This appears to be in degrees with the tow always maintaining a slight upwards pitch. As the tow has a cable running to the surface that would pull it up slightly.

The eleventh column of the data is the towfish's depth above the ocean floor. The unit of depth is meters. As figure 7.20 shows, the towfish appears to have some form of bobbing going on. It is unknown whether this is what happened or if some form of data anomaly occurred.

The twelfth column is the towfish altitude. This is measured in meters and the towfish appears to be attempting to run in some form of constant altitude mode. This is evident in the oscillations around the 30 meter point in figure 7.21.

The final column contains the water depth. This is shown is figure 7.22. This was measured at the position of the tow and measured in meters. The number was computed by adding the tow's depth to the tow's altitude. This is shown best by figure 7.23, which is the difference between the water depth and the tow depth and tow altitude.

7.2 The rawbathy subdirectory

The rawbathy subdirectory contains the raw em1000 files that were generated at the same time the tow was being used. This directory contains the data from lines entitled 5a, 6b, 6c, 6d, 7a, 7b, and 7c. The a, b, c and d are just a means to subdivide the files. Each of these lines consist of multiple files with different

0990570 6 .051101

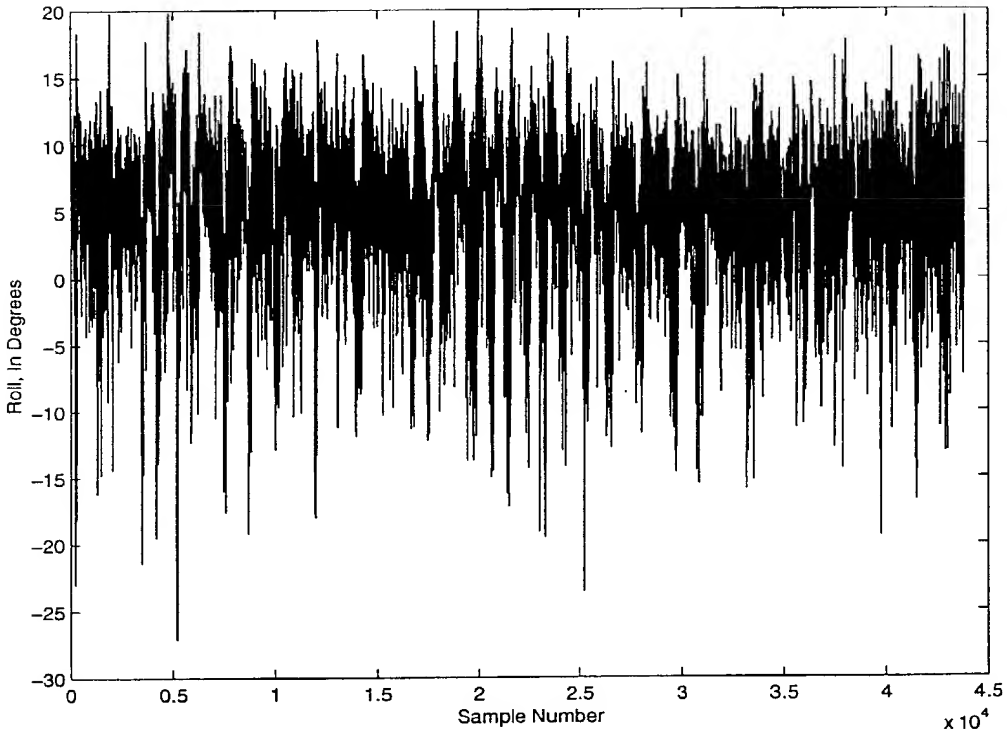

Figure 7.16: The tow's roll, the direction of positive is starboard down.

09905706.051101

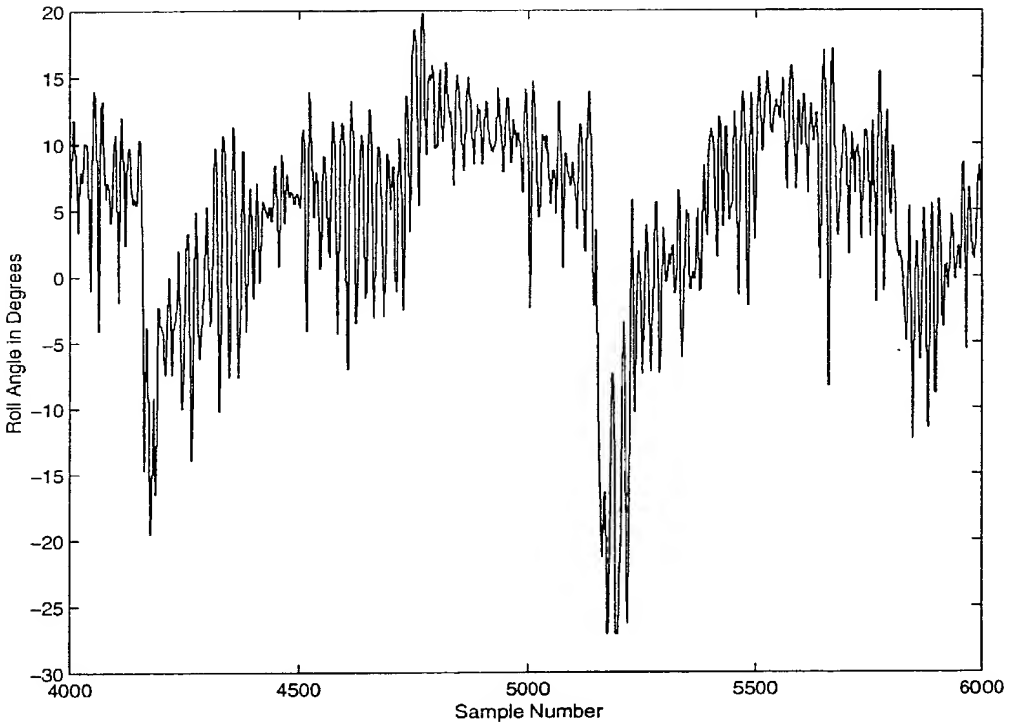

Figure 7.17: The tow's roll, the direction of positive is starboard down.

09905706.051101

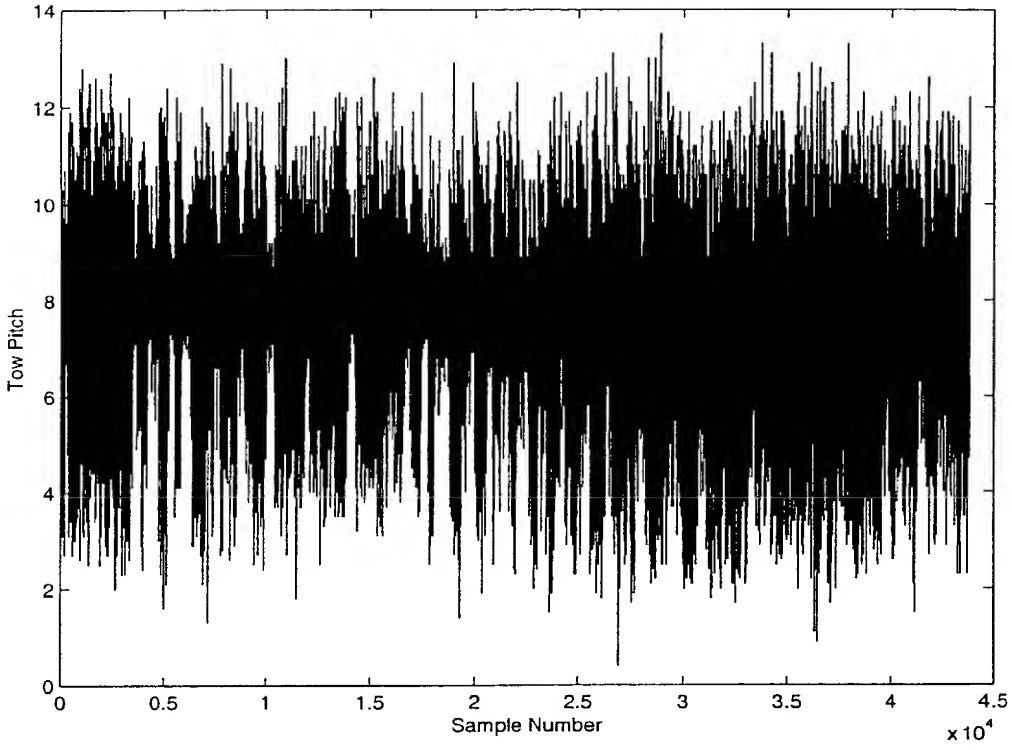

Figure 7.18: The tow's pitch, note the slight up angle at all times.

0990706.051101

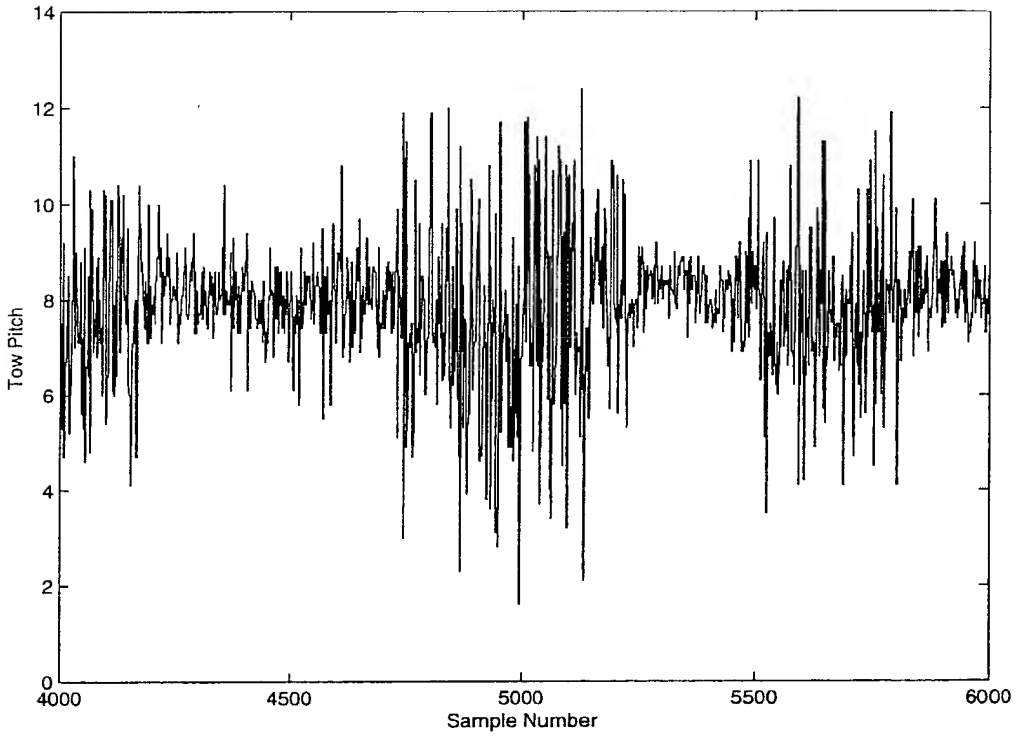

Figure 7.19: The tow's pitch, note the slight up angle at all times.

0990706.051101

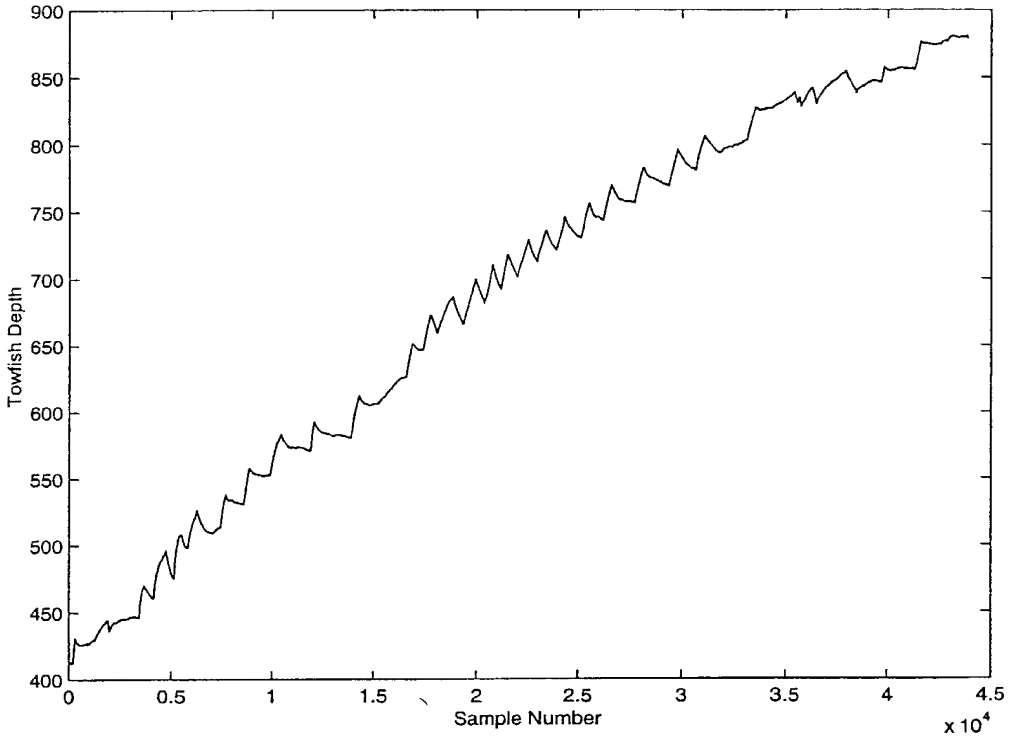

Figure 7.20: The tow's depth in meters.

0990706.051101

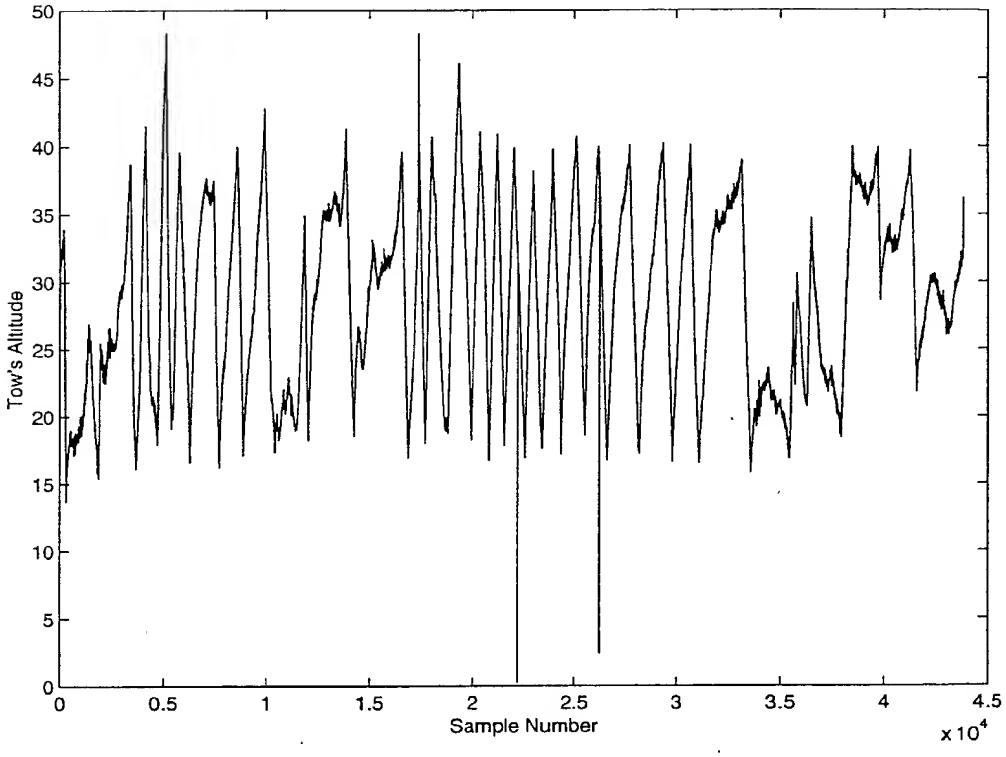

Figure 7.21: The tow's altitude off the bottom in meters.

0990706.051101

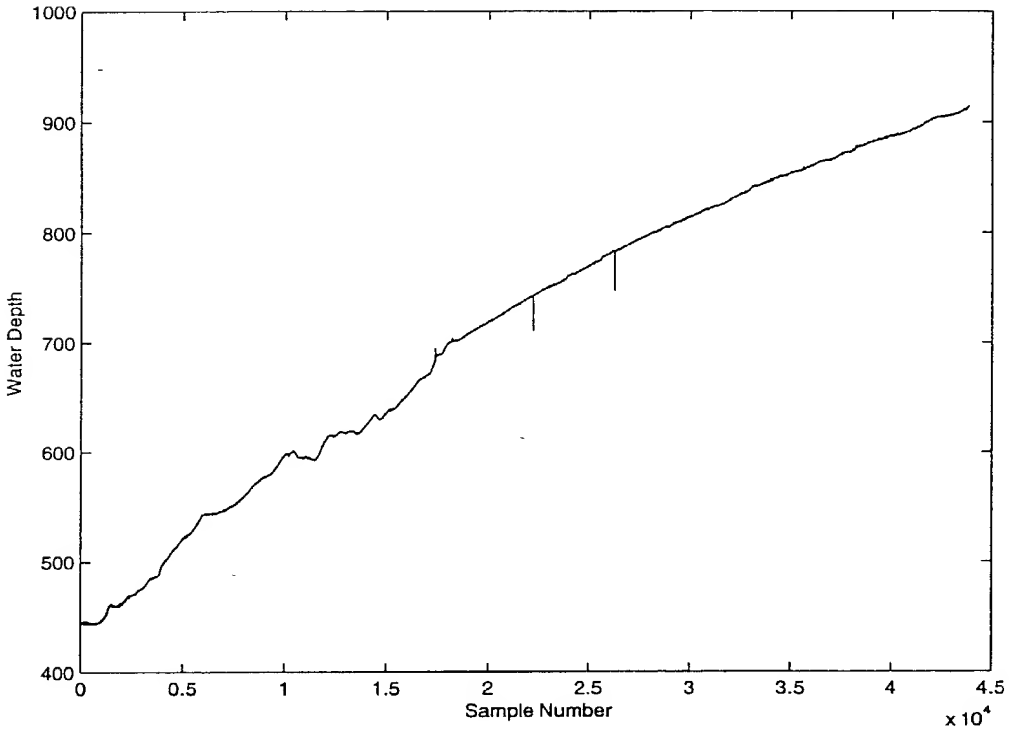

Figure 7.22: The water depth in meters.

0990570606.051101

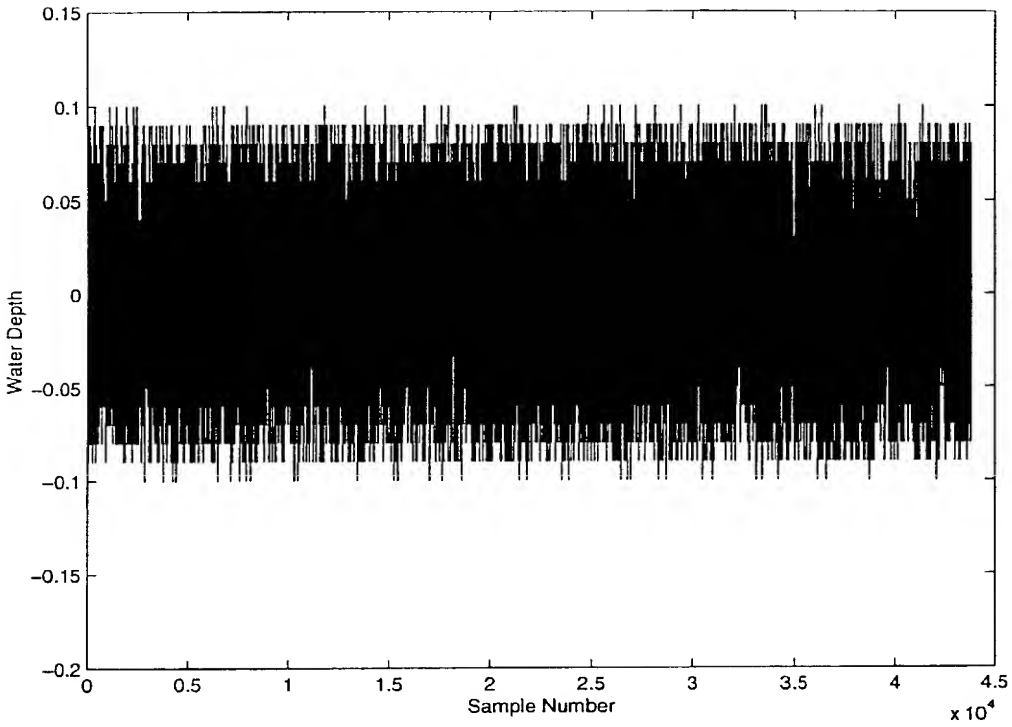

Figure 7.23: The water depth minus the tow's depth and altitude in meters.

extensions. These extensions are compass, em1000, em1000.nav, em1000.med, em1000.processing, em1000.xyz, gps, gps.nav, ascii and rtvelo. Only some of these extensions have been unpackable to date. The entire directory contains 669,646 bathymetric soundings collected by an Simrad 1000 multi-beam echosounder. These points range from 307,503,516 north to 309,064,490 north and 72,626,787 east to 73,216,964 east. A plot of the bathymetric points is included as figure 7.24. The units of the bathymetric points is in the UTM 15 North projection in centimeters. All positions in this directory are in the WGS-84 datum.

This section describes what has been unpacked from each of the file extensions, including representative graphs from line5a. It will describe the bathymetric data coverage for each line. The last part of this section shows the C code used to extract the binary bathymetry into ASCII files.

For the gps file, it contains viewable ASCII NMEA strings. This file does not appear to be necessary because the gps.nav contains the filtered gps positions of the survey vessel. The gps.nav file extension contains four ASCII columns. A sample from this file is included here.

727423.27 3090049.31 916349237.00 -18.08

727423.27 3090049.31 916349237.00 -18.08

727423.27 3090049.31 916349237.00 -18.08

The first column appears to be the survey ship's east position. The units are meters. The east position is shown vs. sample as figure 7.25.

The second column is the filtered gps north position. The survey vessels north position is shown as figure 7.26. This is also in meters, and has been independently confirmed.

The track of the survey vessel in line5a is shown as figure 7.27. This track is consistent with the line5a bathymetry which will be shown later.

09905706.051101

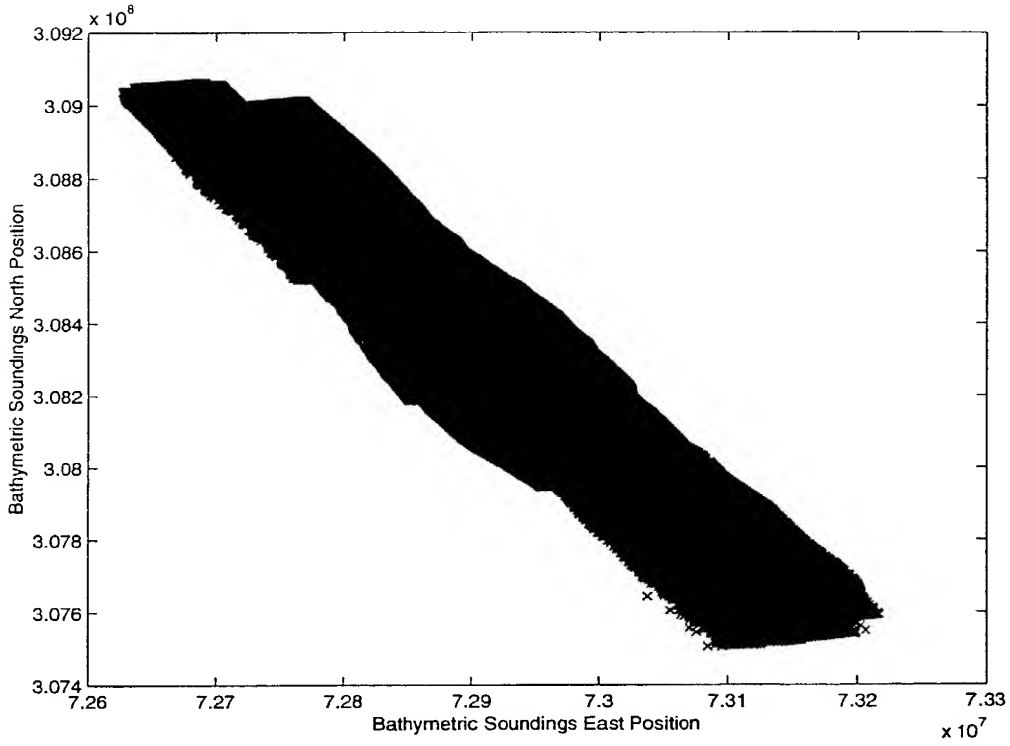

Figure 7.24: The entire raw bathymetric data set.

0990570G.051101

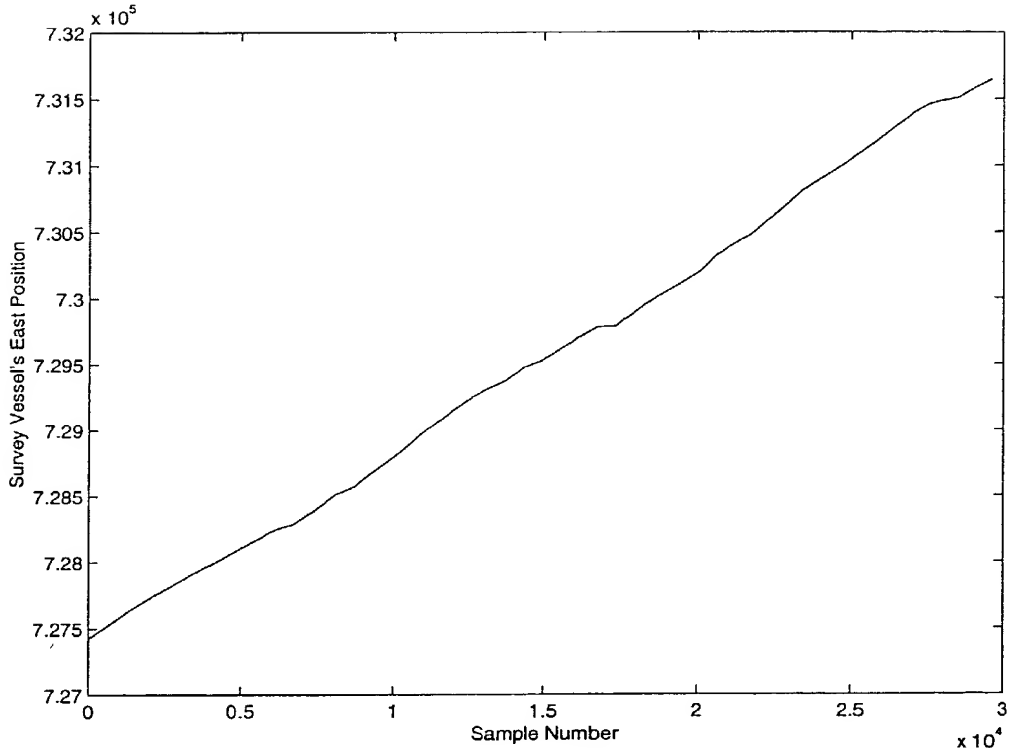

Figure 7.25: The survey vessel's filtered gps east position in meters vs. sample.

09905706.051301

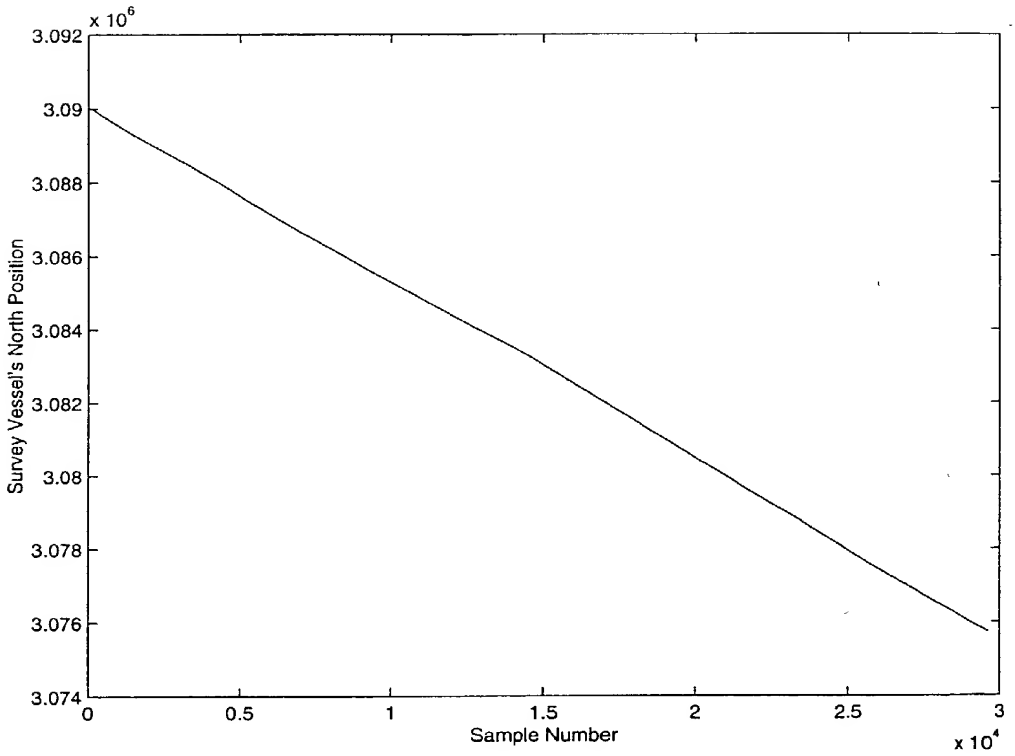

Figure 7.26: The survey vessel's filtered gps north position vs. sample.

09905706.051101

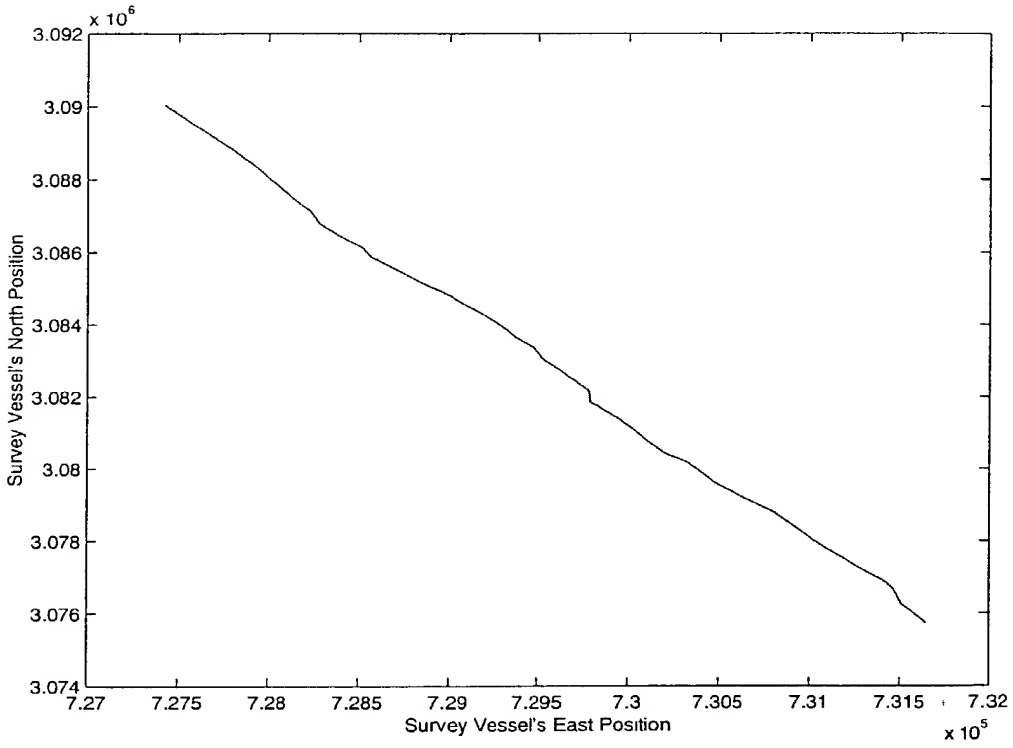

Figure 7.27: The survey vessel's filtered gps track.

0990706.051101

The third column is the time reference. This conclusion is reinforced by examining the linear nature of the graph, figure 7.28. There is a single dropout, which is not unexpected for gps. The unit is seconds measured from midnight on Jan. 1, 1970.

The forth column is to be the vessel's altitude with respect to the geoid. The -18.08 meters is the difference between geoid in this area. This data tells us that the bathymetry is measured with respect to mean sea level. The unit is meters. The plot is shown as figure 7.29.

The files ending in `em1000` are Simrad internal files that are not needed. The `em1000.med}` are median filtered bathymetry points however, the internal fo: contain the raw bathymetry for each line segment. These files will be discussed more later. The `rtvelo` files contain three columns in ASCII. The first column is water depth, the second is sound velocity at that depth and the third is the Harmonic mean. The `em1000.processing` is a C&C internal use file that is not related to LOST2's needs. The `compass` file contains two columns, the first of which is time and the second is raw NMEA strings.

The `em1000.xyz` files contained the bathymetry for each line segment. These files were extracted using the C program `bintoascii.c` into the `.ASCII` files. The next few paragraphs briefly describe the bathymetry. A sample from the line5a.ascii file in included here.

72712937 308995707 -47189

72712889 308996386 -46979

72712982 308996460 -46938

72712890 308996833 -46939

The first line segment is line5a. An x-y plot of the data is shown as figure 7.30.This line contains 238,457 bathymetry points, ranging from 307,560,712 cm

09905706.051101

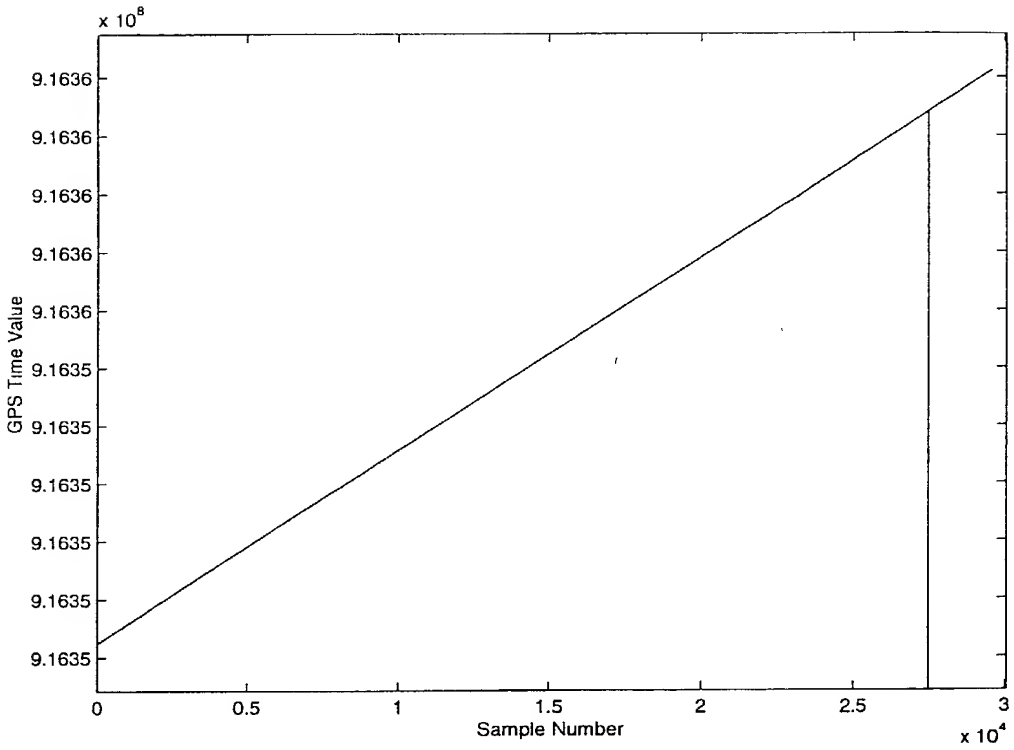

Figure 7.28: The gps time value vs. sample.

0990706.051101

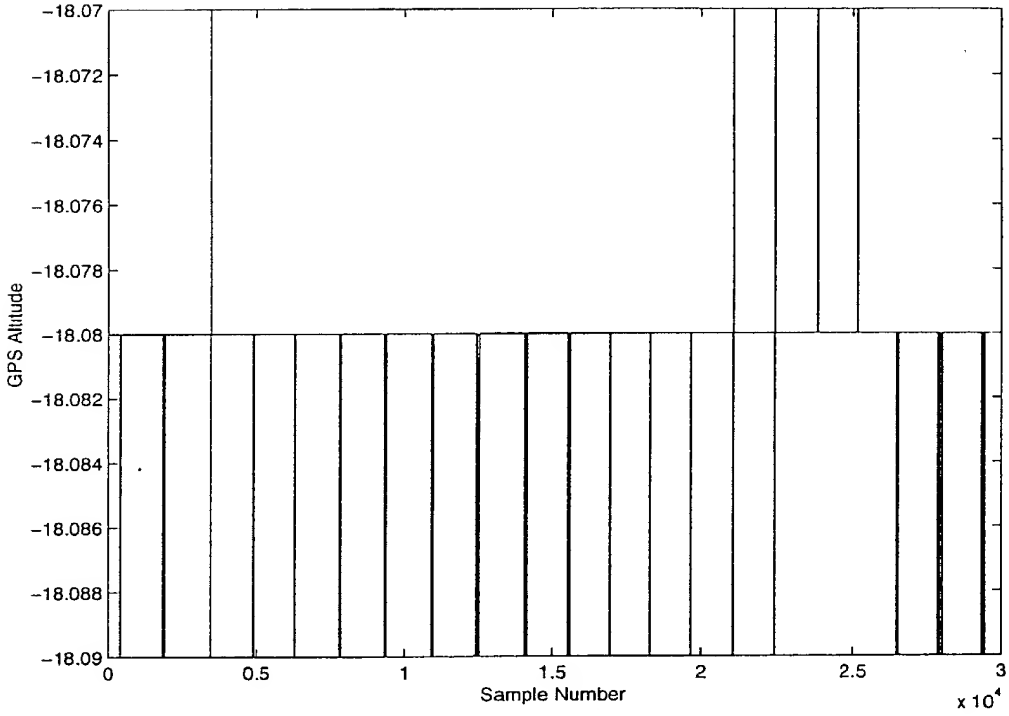

Figure 7.29: The survey vessel's filtered gps altitude vs. sample.

north to 309,014,377 cm north. The east values range from 72,712,626 cm to 73,216,964 cm. The depth in this area ranges from -93808 cm to -45938 cm.

The next line segment is line6b. An x-y plot of the data is shown as figure 7.31. This line contains 15,832 bathymetry points, ranging from 308,584,515 cm north to 308,679,445 cm north. The east values range from 72,774,655 cm to 72,876,042 cm. The depth in this area ranges from -66684 cm to -54726 cm.

The next line segment is line6c. An x-y plot of the data is shown as figure 7.32. This line contains 72,895 bathymetry points, ranging from 308,274,394 cm north to 308,618,662 cm north. The east values range from 72,796,720 cm to 72,965,832 cm. The depth in this area ranges from -79291 cm to -58691 cm.

The last line 6 line segment is line6d. An x-y plot of the data is shown as figure 7.33. This line contains 105,218 bathymetry points, ranging from 307,503,516 cm north to 308,312,389 cm north. The east values range from 72,870,457 cm to 73,216,964 cm. The depth in this area ranges from -102619 cm to -67471 cm.

The first line 7 line segment is line7a. An x-y plot of the data is shown as figure 7.34.This line contains 132,466 bathymetry points, ranging from 308,448,002 cm north to 309,064,490 cm north. The east values range from 72,626,787 cm to 72,881,520 cm. The depth in this area ranges from -70346 cm to -42022 cm.

The next line 7 line segment is line7b. An x-y plot of the data is shown as figure 7.35. This line contains 56,449 bathymetry points, ranging from 308,183,389 cm north to 308,467,195 cm north. The east values range from 72,797,190 cm to 72,948,919 cm. The depth in this area ranges from -79117 cm to -61005 cm.

The last line segment is line7c. An x-y plot of the data is shown as figure 7.36. This line contains 48,329 bathymetry points, ranging from 307,795,188 cm north to 308,210,341 cm north. The east values range from 72,860,119 cm to 73,102,536 cm. The depth in this area ranges from -88194 cm to -77234 cm.

09905706.051101

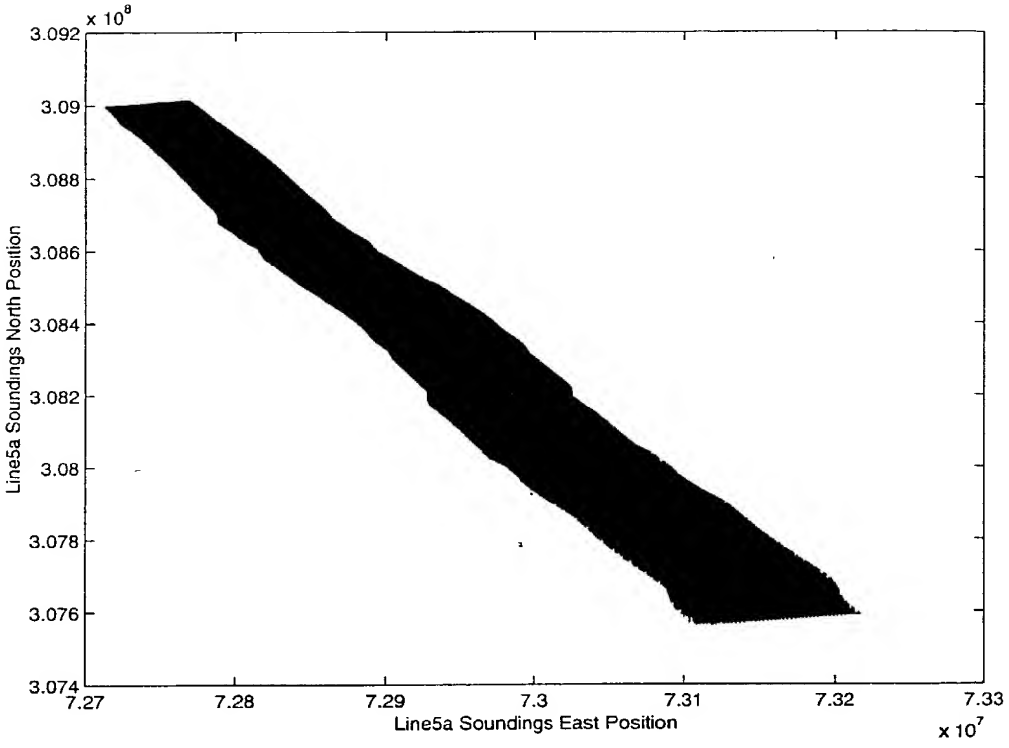

Figure 7.30: Line 5a's x-y plot of the bathymetry.

0990706.051101

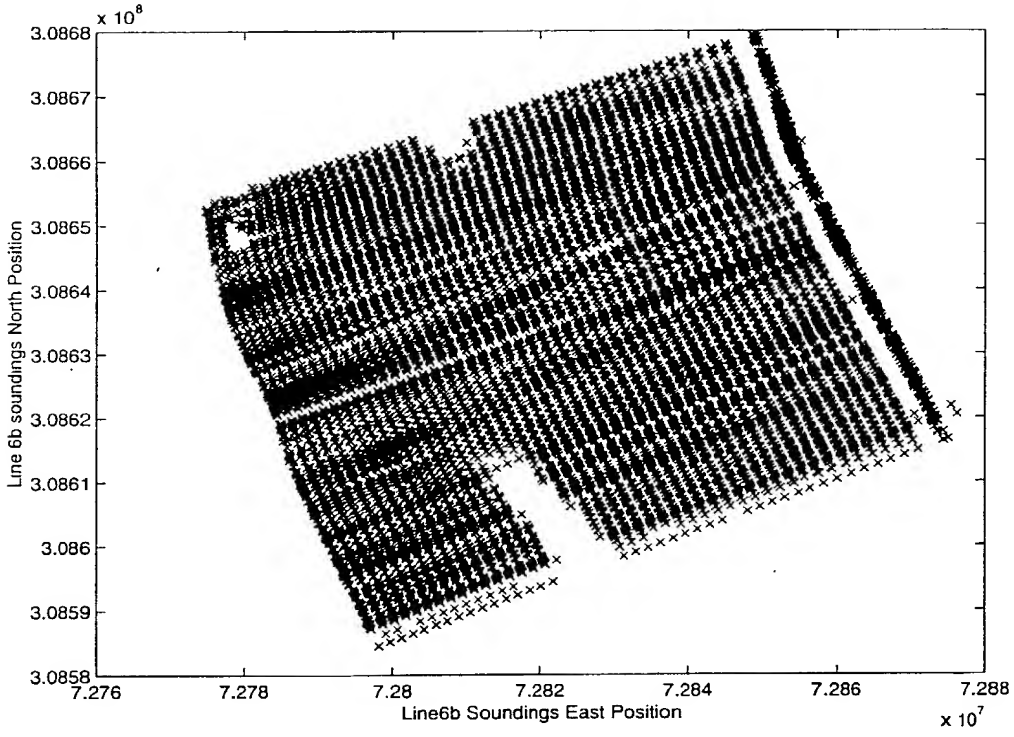

Figure 7.31: Line 6b's x-y plot of the bathymetry.

09905706.051101

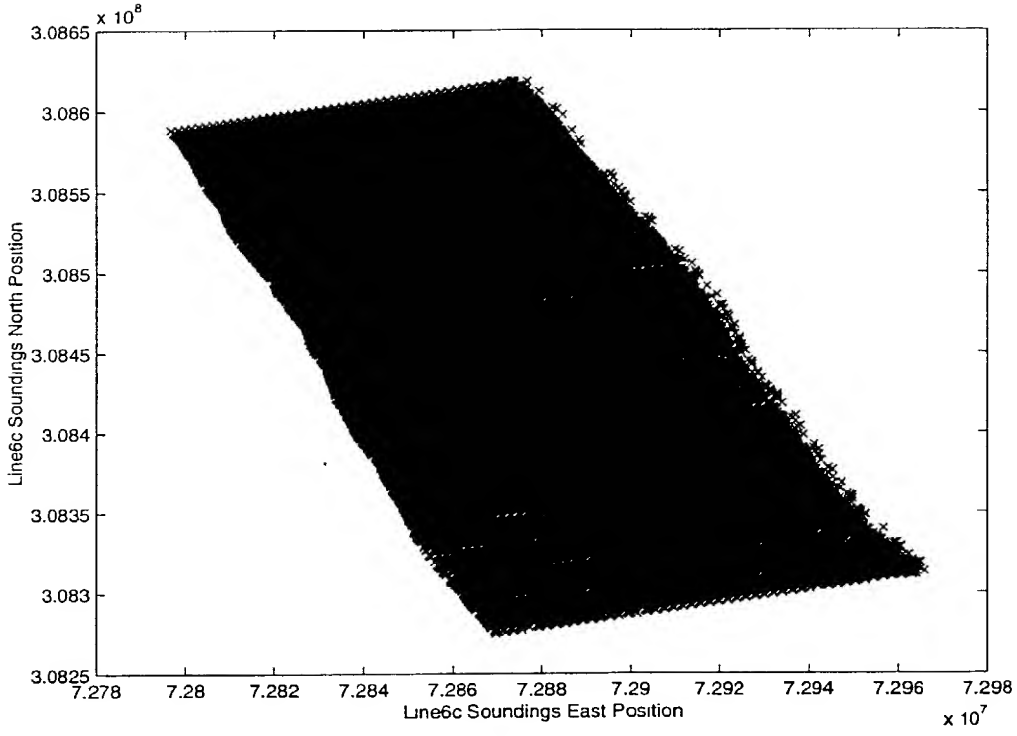

Figure 7.32: Line 6c's x-y plot of the bathymetry.

09905706.051101

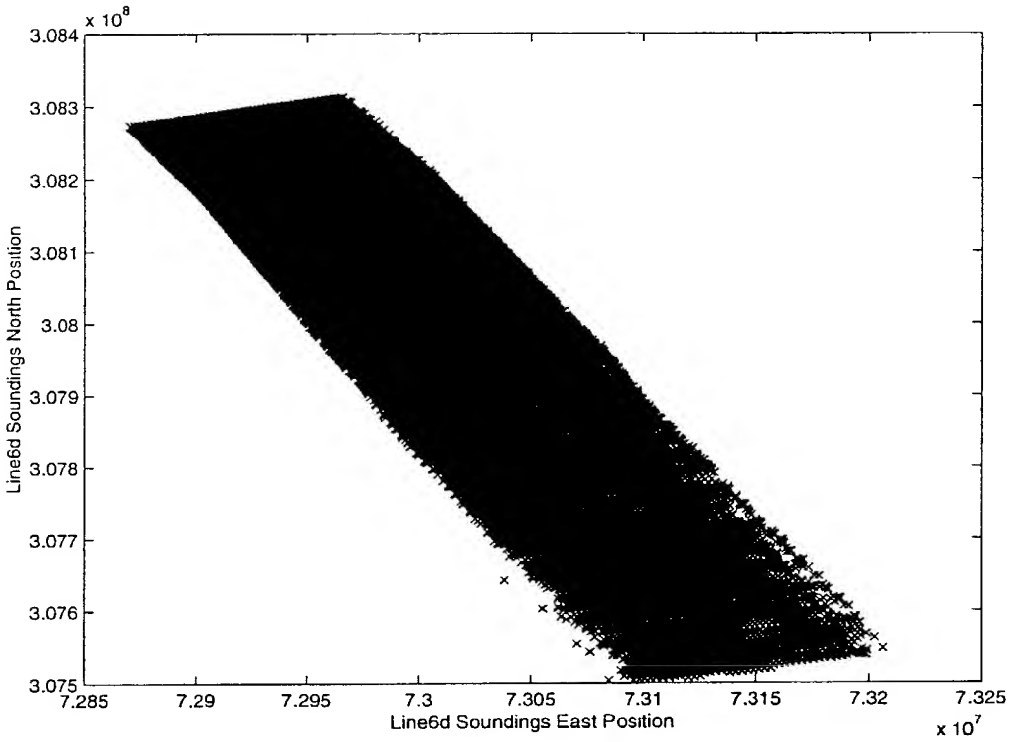

Figure 7.33: Line 6d's x-y plot of the bathymetry.

0980706.051101

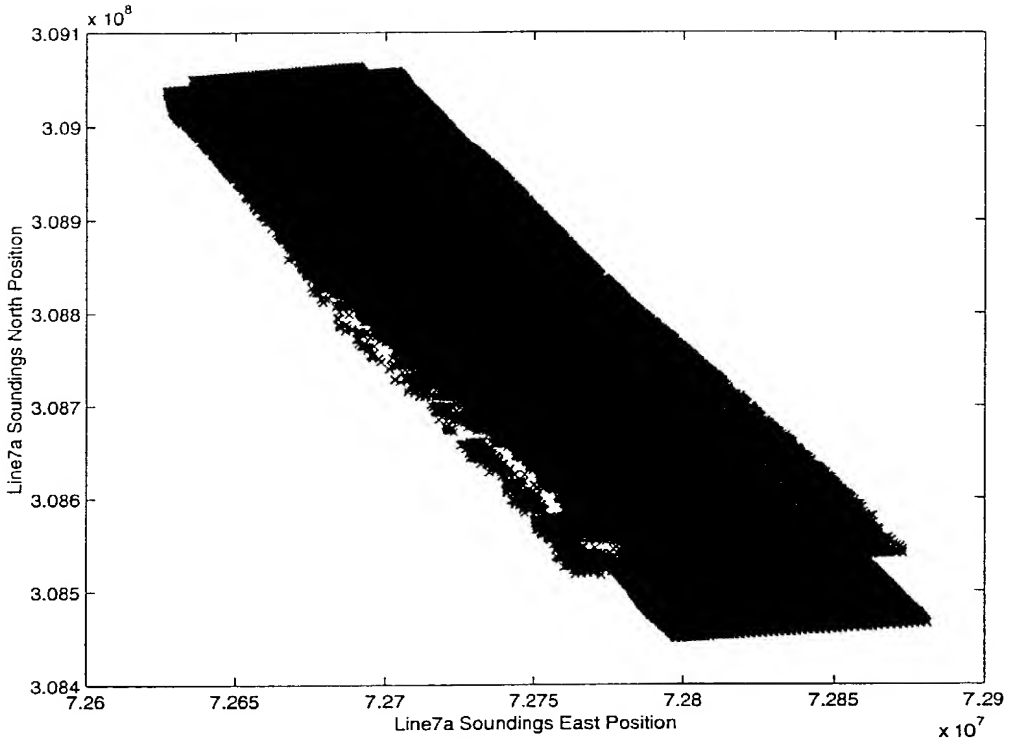

Figure 7.34: Line 7a's x-y plot of the bathymetry.

09905706.051101

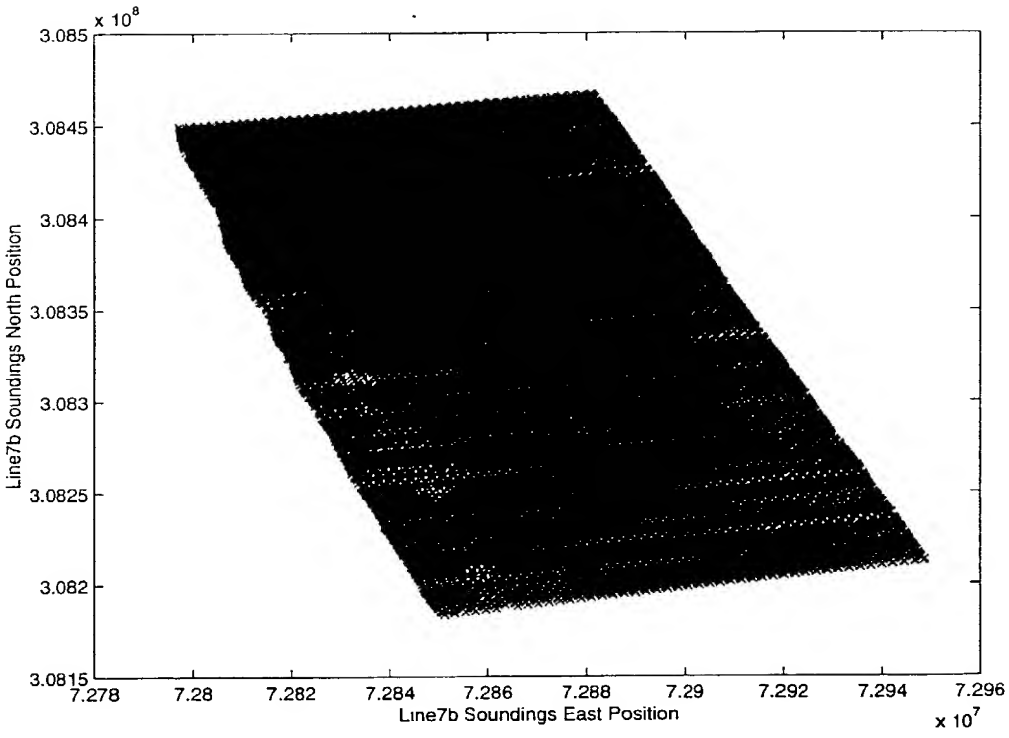

Figure 7.35: Line 7b's x-y plot of the bathymetry.

09905706.051101

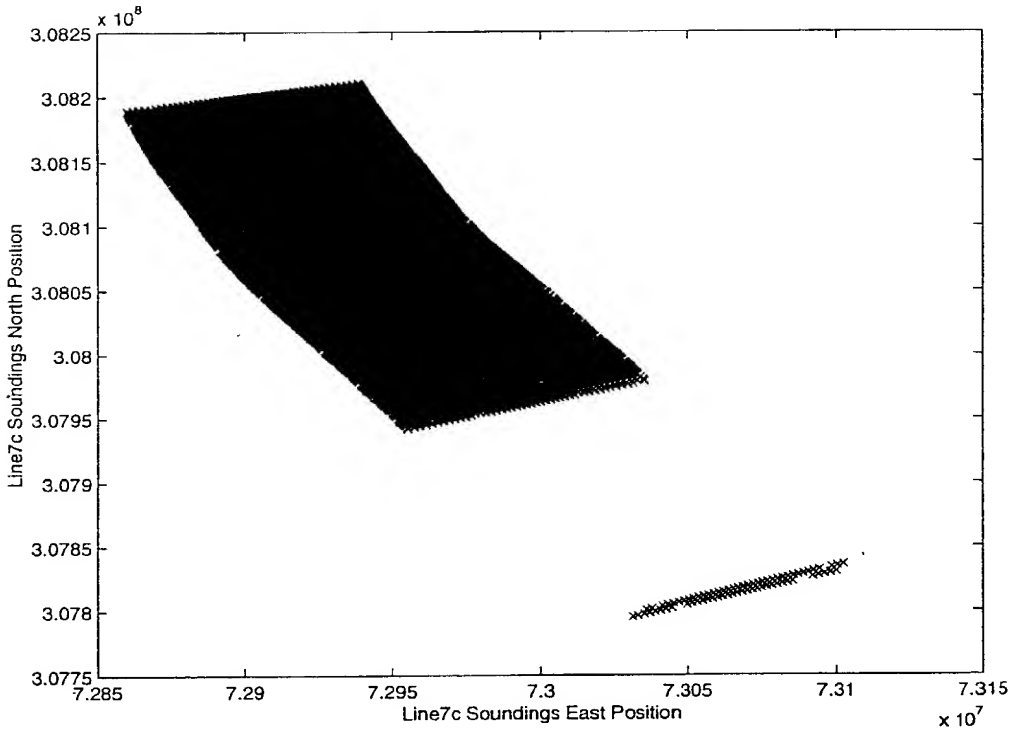

Figure 7.36: Line 7c's x-y plot of the bathymetry.

D9905706.051101

Displaying all of the line files on a single colorized graph shows the coverage from each line file. The combination is shown as figure 7.37.Each of the different colors represents a different line file.

The only remaining processing to be done on the raw bathymetry is gridding. This will allow for a surface plot to be constructed so that the effect of terrain on performance can be determined.

The last part of this section is the C code used to extract the binary data. Because this data was created on a Sun workstation and Sun's are bigendian and padded so that all storage is a multiple of four bytes, this program must be run on a Sun workstation. The ASCII files it creates, however, are completely portable to any system.

```
#include<stdio.h>
/*converts binary to ASCII*/

int main(void){
FILE *fp1,*fp2;
unsigned x;
int count,count1;

fp1=fopen("line5a.0.em1000","rb");
fp2=fopen("line5a.ascii","w");

count=0;
count1=0;
while((fread(&x,sizeof(unsigned),1,fp1)==1){
fprintf(fp2,"%u ",x);
```

09905706.051101

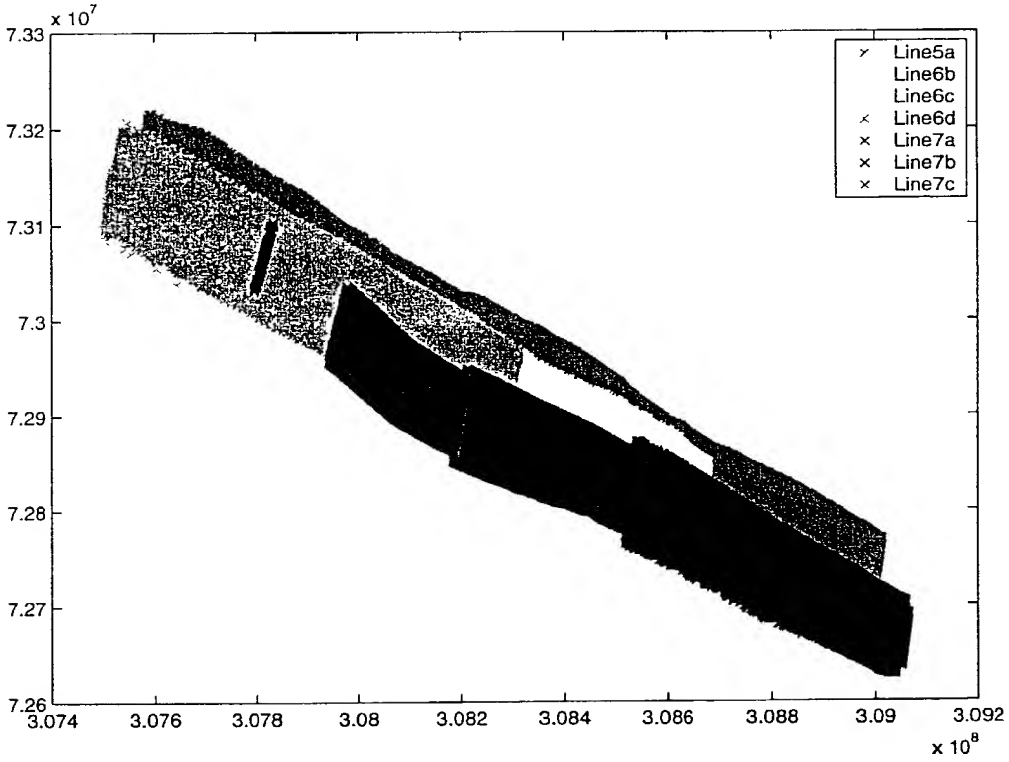

Figure 7.37: The plot of the bathymetry, showing each line's coverage.

0990706.051101

```
count++;
count1++;
if (count==3){
fprintf(fp2,"\n");
count=0;}
}

fclose(fp1);
fclose(fp2);
return(0);}
```

7.3 Complete File Listing

What follows is the complete file listing of the `towfishcc` directory. This listing contains all the files either received from C&C or created in house to manipulate the data.

```
towfishcc:
fishdata/
rawbathy/

towfishcc/fishdata:
LINE5.txt
LINE6.txt
LINE7.txt
line5.txt
```

towfishcc/rawbathy:

b.tmp

bathy.ascii

bathy.ascii1

bathy.ascii2

bathy.ascii3

bathy.ascii4

bathy.ascii5

bathy.ascii6

bathy.com

bathy.txt

bintoascii*

bintoascii.c

line5a.0.compass

line5a.0.em1000

line5a.0.em1000.med

line5a.0.em1000.nav

line5a.0.em1000.xyz

line5a.0.gps

line5a.0.gps.nav

line6b.0.compass

line6b.0.em1000

line6b.0.em1000.med

line6b.0.em1000.nav

line6b.0.em1000.xyz

line6b.0.gps

line6b.0.gps.nav

line6b.0.rtvelo

line6c.0.compass

line6c.0.em1000

line6c.0.em1000.med

line6c.0.em1000.nav

line6c.0.em1000.xyz

line6c.0.gps

line6c.0.gps.nav

line6c.0.rtvelo

line6d.0.compass

line6d.0.em1000

line6d.0.em1000.med

line6d.0.em1000.nav

line6d.0.em1000.xyz

line6d.0.gps

line6d.0.gps.nav

line6d.0.rtvelo

line7a.0.compass

line7a.0.em1000

line7a.0.em1000.med

line7a.0.em1000.nav

line7a.0.em1000.processing

line7a.0.em1000.xyz

line7a.0.gps

line7a.0.gps.nav

line7a.0.rtvelo

line7b.0.compass

line7b.0.em1000

line7b.0.em1000.med

line7b.0.em1000.nav

line7b.0.em1000.processing

line7b.0.em1000.xyz

line7b.0.gps

line7b.0.gps.nav

line7b.0.rtvelo

line7c.0.compass

line7c.0.em1000

line7c.0.em1000.med

line7c.0.em1000.nav

line7c.0.em1000.processing

line7c.0.em1000.xyz

line7c.0.gps

line7c.0.gps.nav

line7c.0.rtvelo

0990706.051101

Chapter 8

Results Using Real Data

Using the data supplied by C&C Technologies, Inc. that was described in the previous chapter, the LOST2 system was tested and found to produce similar positions as those provided by the USBL system on each track. There were four preprocessing steps that had to be performed before the system could be run over the data, which are described in section 8.1. The results of the system on the three lines are then presented in section 8.2.

8.1 Processing Required on the Data

There were four main problems with the data that had to be corrected before the system could be run. 1) The ground truth positions provided by the USBL system were not usable in their original form. 2) Because the towfish did not have a velocity sensor it was necessary to extract this input from the data streams and then pass it to the system. 3) It was discovered that there was a bias in the depth data provided by the towfish, which as is shown in section 8.2 had to be removed before accurate results were provided. 4) The final preprocessing stage required the slant range to be backed out from the ground truth positions.

0990570606.051101

8.1.1 Preprocessing Performed on the Ground Truth

The ground truth data had several problems that had to be corrected. The first was that it was collected in different units as compared to the rest of the data. The second was that is was stored in reverse order from the standard method of storing UTM positions. The final correction required the resampling of the data to synchronize the system clocks.

The first problem that was discovered was the units of the position were incorrect. After talking with C&C, it was discovered that this data was collected by the Winfrog software package, which records its positions in feet not meters. This was corrected by multiplying the positions by correct conversion factor. This translated the positions into meters.

Once the units were corrected for, the positions still did not line up with the GPS ship data. After further consultation with C&C, it was discovered that the positions were reversed. After reversing the order of the position points, the positions then did reasonability line up with the data.

The final problem that had to be corrected was the fact that the USBL data was sampled five times as fast as the rest of the data. This was corrected by decimating the data by a factor of five. The standard approach was taken to do this, first building an 11 point FIR lowpass filter to prevent aliasing in the result by reducing the spectrum by a factor of five and then removing four out of five samples[57]. The filtering had the added benefit of smoothing the ground truth data to produce more realistic results, but this was not why the filtering was done. The line5 track after all preprocessing is shown as figure 8.1.

0990706.051101

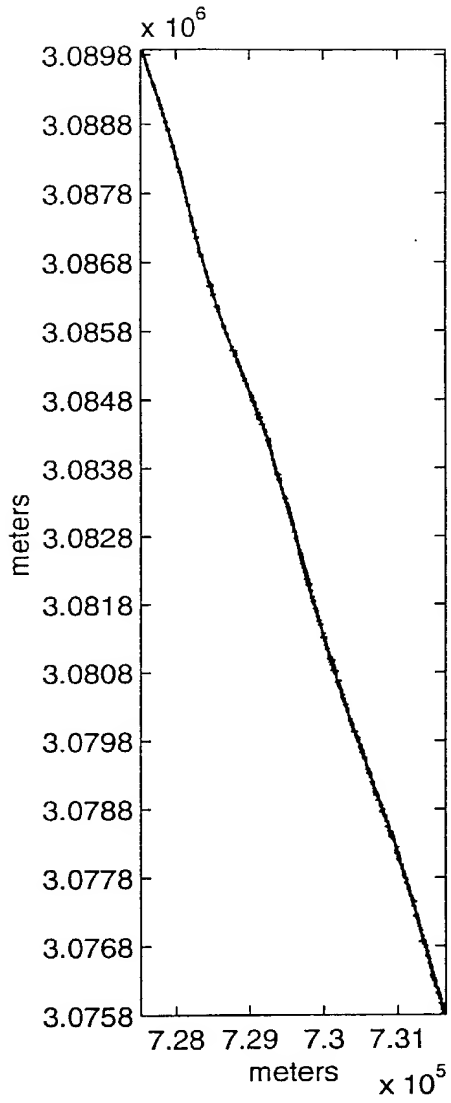

Figure 8.1: The resampled version of the line5 data

0990706.051101

8.1.2 Velocity Generation

One of the required inputs that the LOST2 system requires is the velocity of the vessel. However, the towbody used to collect this data was not equipped with a reliable velocity sensor, so this data had to be synthetically generated. The towing vessel's GPS positions were available and were used to generate the velocity.

The velocity input was generated by simply differencing adjacent samples from the GPS system and assuming this was the velocity. This was done for several reasons. First the GPS position was a measured quantity and so it was not a simulated input. Secondly due to the nature of the towing operation, the ship's velocity is not quite the towbody's velocity, adding an unmodled velocity error into the system. This is desirable because any sensor is going to have inaccuracies in it. Finally, this has the required units for the LOST2 system of meters since the last sample.

8.1.3 Bias Removal

It was noticed early in the processing that the measured depths at theUSBL positions and did not match the bathymetry. This causes a large problem because one of the major assumptions made by the LOST2 system is that the depth measurements are accurate. So the bias had to be computed and removed.

This was accomplished via nearest neighbor analysis. The closest bathymetry point was found and the depths were then differenced. This number was then averaged with all the other points on the run and this was the computed bias. This method assumed that the bias was not time varying and appears to be a correct assumption. The bias for the runs were 2.4 meters, 2.65 meters and 2.58 meters for lines 5, 6 and 7 respectively. The RMS error on these numbers when

0990706.051101

compared to the nearest bathymetry point were 2.15 meters, 2.16 meters and 2.26 meters respectively.

8.1.4 Slant Range Generation

The other input that the LOST2 system requires is the slant range to the known point. Due to the fact that there was no slant range transponder present, its output had to be simulated. This is the weakest part of the system validation tests.

The slant range was generated by computing the distance from the towing vessel to the filtered USBL vessel position estimate. A random variable generated from a Normal (0,4) distribution was then added to this result to simulate measurement error. This sum was called the slant range and then input into the LOST2 system as the final required input.

8.2 Results

The results from the sea trials are presented in two sets. The first set of tests presents the results from all three lines, both with and without the bias in the sensors removed. The second section describes some of the system behaviors that were noted during the testing phase of this project.

Before any testing could be done, there are three parameters that must be supplied to the LOST2 system. The first is the intial estimate of the vessels position. This is generated by projecting the towing ship's position back by the slant range. This provides the best estimate of position before any external positioning can be performed. The other two inputs are the heading which was taken from the heading sensor provided by C&C and the number of samples to be positioned, which is determined by the length of the files.

0990706.051101

8.2.1 Results for Each Track

This section presents the results for each track. The results for each line are presented in graphical form with the doted line being the USBL position and the solid line being the LOST2 predicted track. The tracks are presented in the order of the whole run with the bias present, a zoomed in version of the run with bias, the run with the bias removed and the zoomed in version with the bias removed.

The first line presented is the results from line 5. The version with the bias of 2.4 meters shallow present is shown as figure 8.2. As is evident in this track and in figure 8.3 the predicted location is to the right of the real track. This is expected because on line 5 the bathymetry in the area get shallower as it goes from right to left, and therefor a deep bias will shift the position to right. After removing the bias the path tracks closely as is shown in figure 8.4. The path lays within the envelope of the USBL position as is shown in figure 8.5. These results show that the system works on line 5.

The next line is Line 6. Line 6 has the largest bias and therefore is offset more than the other runs. The version with the bias of 2.65 meters shallow present is shown as figure 8.6. As is evident in this track and in figure 8.7 the predicted location is to the left of the real track. This is expected because on line 6 the bathymetry in the area gets deeper as it goes from right to left, and the deep bias would shift the position to the left. After removing the bias the path tracks closely as is shown in figure 8.8. The path lays within the envelope of the USBL position as is shown in figure 8.9. These results show that the system works also works on line 6.

The final run was Line 7. Line 7, although is had a smaller bias, had a larger variance in the accuracy of the depth measurement. The results of this run show that the position without removing the bias is shift to the left, as shown in

09905706.051101

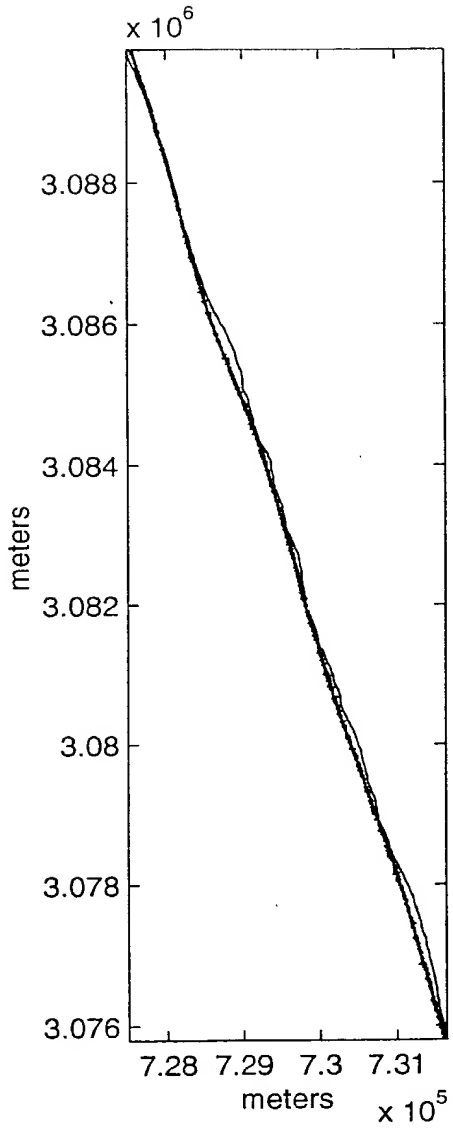

Figure 8.2: Line 5 with the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

09905706.051101

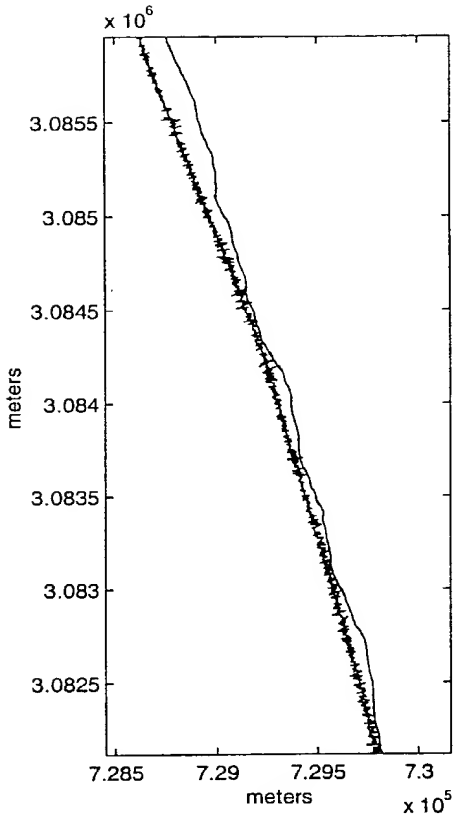

Figure 8.3: A close up of line 5 with the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

09905706.051101

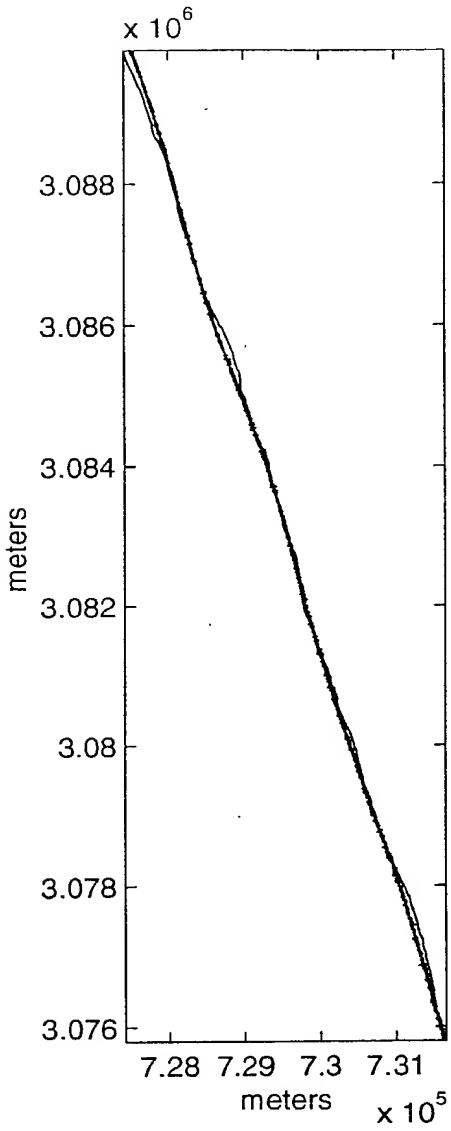

Figure 8.4: Line 5 without the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

0990706.051101

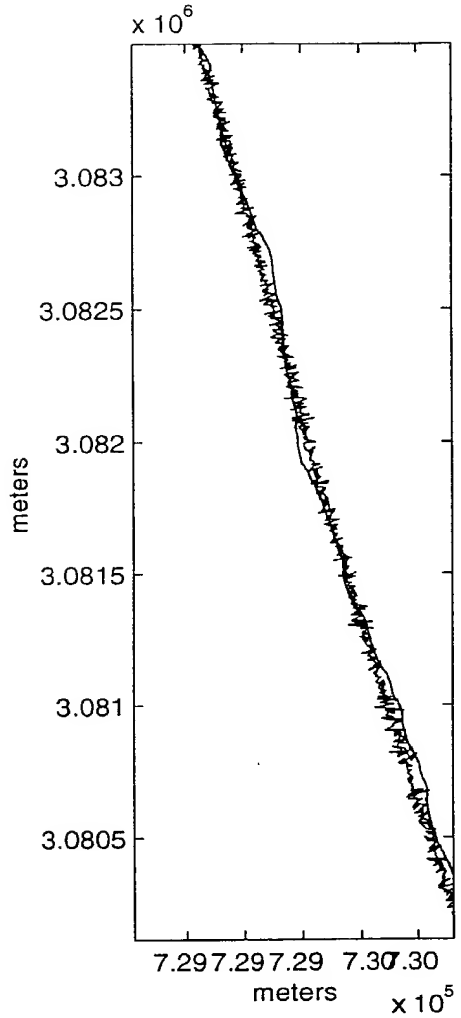

Figure 8.5: A close up of line 5 without the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

09905706.051101

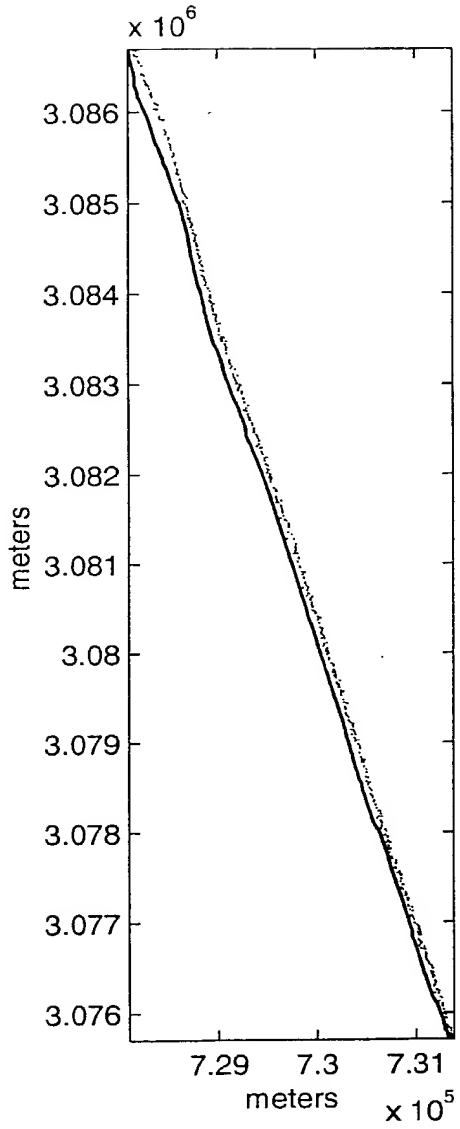

Figure 8.6: Line 6 with the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

09905706 .051101

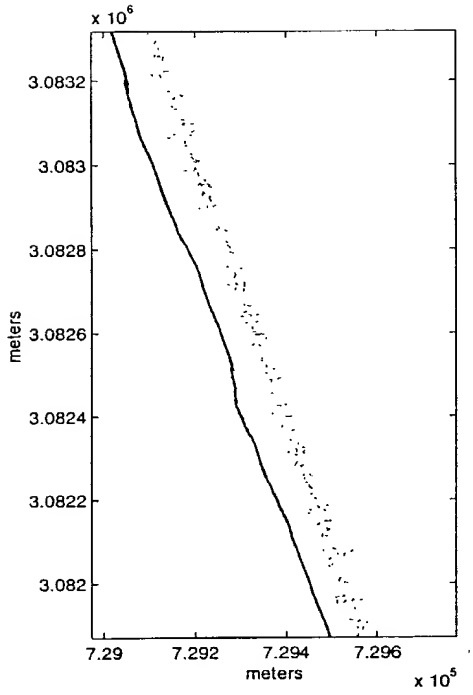

Figure 8.7: A close up of line 6 with the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

09905706.051101

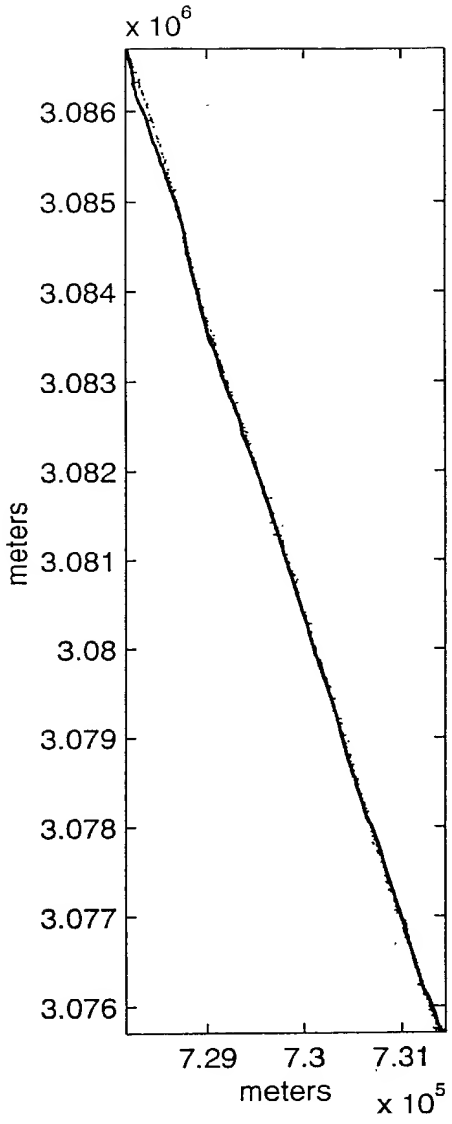

Figure 8.8: Line 6 without the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

09905706 .051101

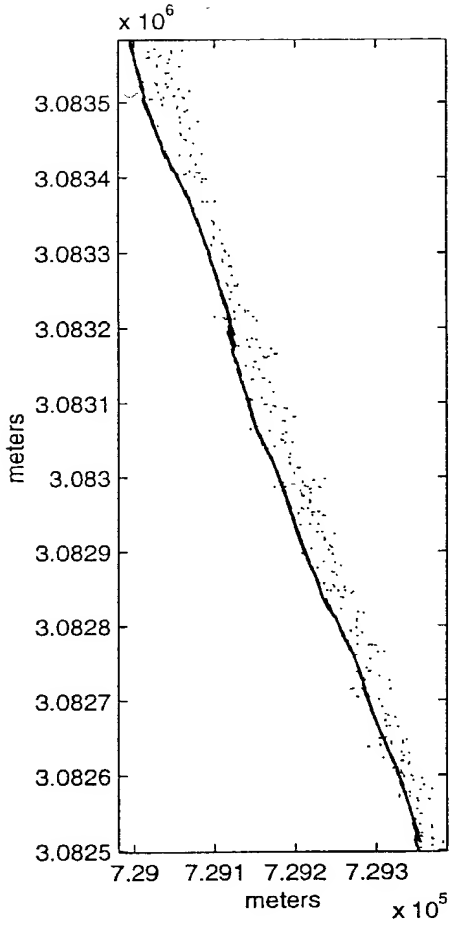

Figure 8.9: A close up of line 6 without the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

9990706.051101

figures 8.10 and 8.11. This was again shifted in the expected direction due to the depth profile of the bathymetry. However, once the bias was removed there was a problem in acquiring the path. This is shown in the wild oscillations at the top of figure 8.12. The reason for this is unknown but that is where the largest errors in the measured depth are so this is the suspected reason for this behavior. The result does track once it acquires though as shown in figure 8.13.

The results of the trackss prove that given an accurate depth measurement and a high-enough resolution bathymetric data, LOST2 provides accurate position estimates. These results also have lead to some other notable behaviors that are discussed in the next section.

8.2.2 System Behaviors

There were two other interesting behaviors noted during the system tests. The first and most expected was the presence of the startup transient. The second behavior is a tendency to at times wander away from the correct path and then regain the correct path.

The first behavior that is noted is the startup transient present in each track. These transients are about 15 minutes in length during which the system closes the error from over 100 meters in most cases, due to the nature of the system initialization, to within the envelope of the USBL positioning system. Line 5's startup transient is shown as figure 8.14.

The other behavior that is noticed is the tendency to lose lock on the correct path for a short period of time. This most often occurs in flatter spots in the bathymetry in which the depth provides less information on position than is areas with steeper contours. This also can occur when the velocity estimate pulls the vessel away from the correct path. An example of this condition, the

0990706.051101

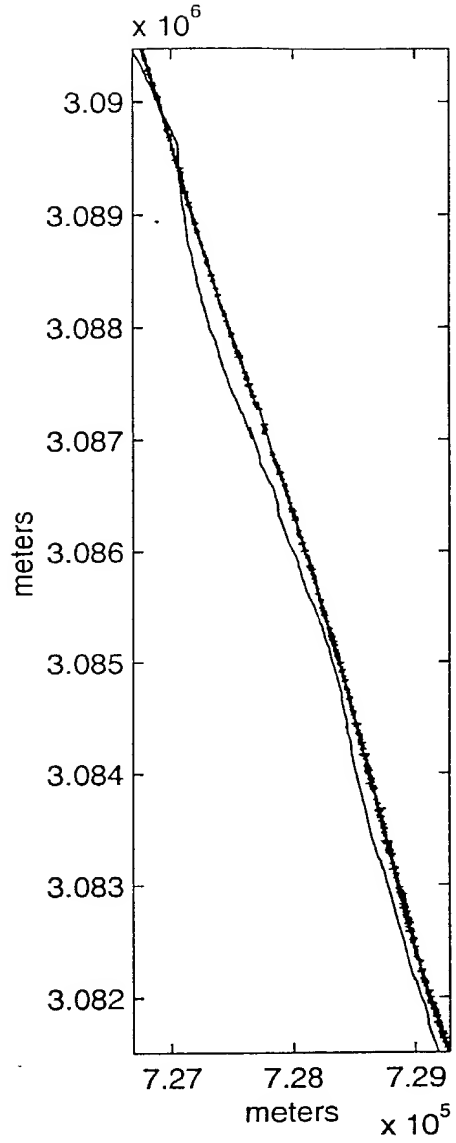

Figure 8.10: Line 7 with the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

0990706.051101

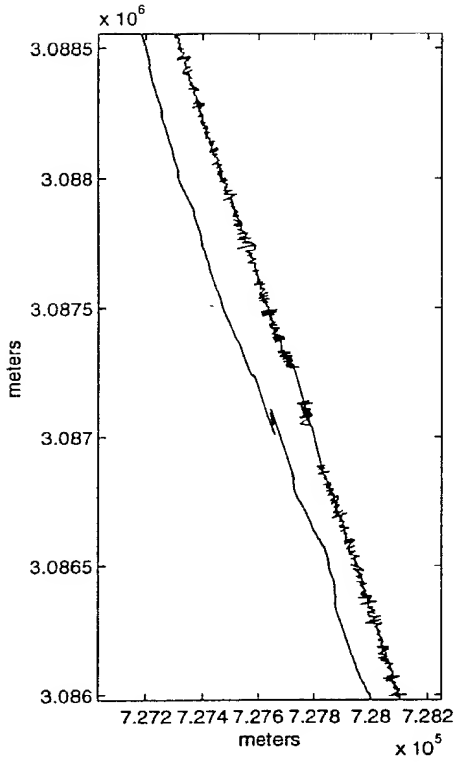

Figure 8.11: A close up of line 7 with the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

09905706.051101

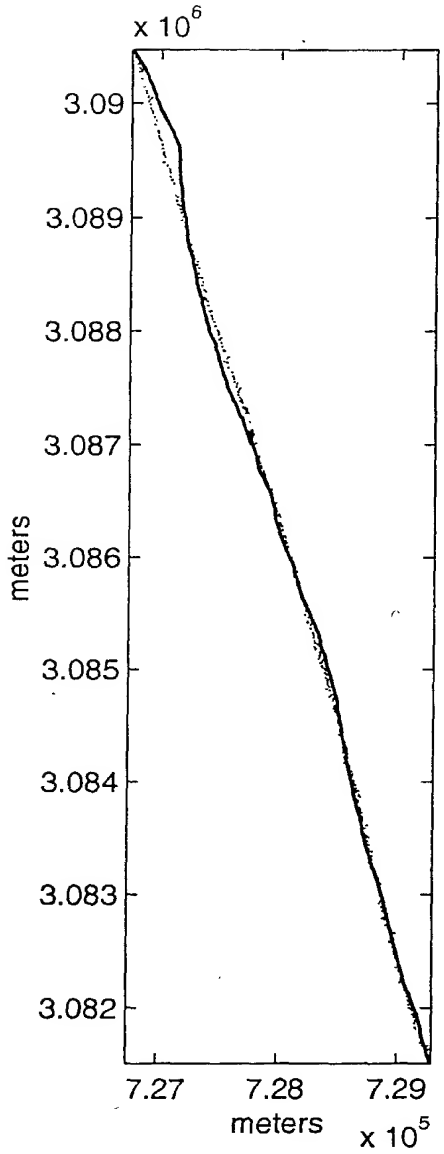

Figure 8.12: Line 7 without the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

0990706.051101

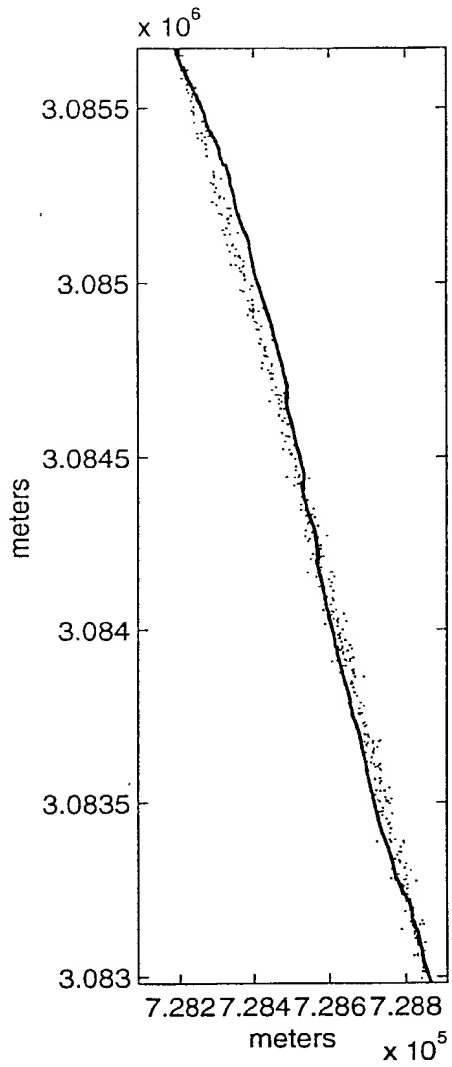

Figure 8.13: A close up of line 7 without the Bias present. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

0990706.051101

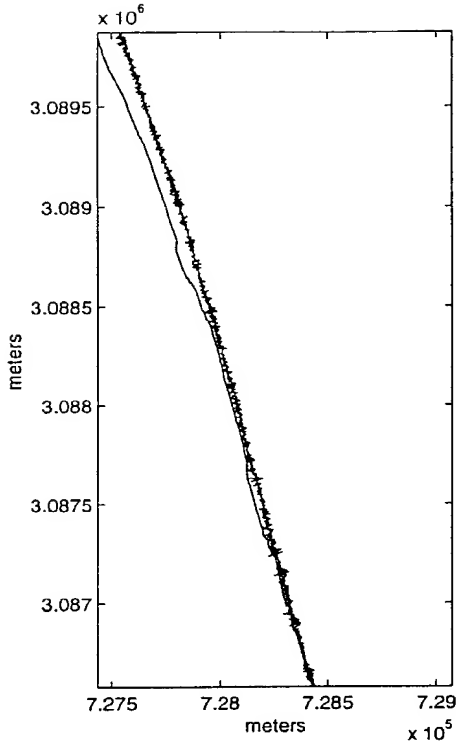

Figure 8.14: A close up of line 5's startup transient. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

0990706.051101

exact cause of which is unknown, is shown in figure 8.15. However even when the system loses the correct path it eventually reacquires the correct path, showing that the system is robust.

09905706.051101

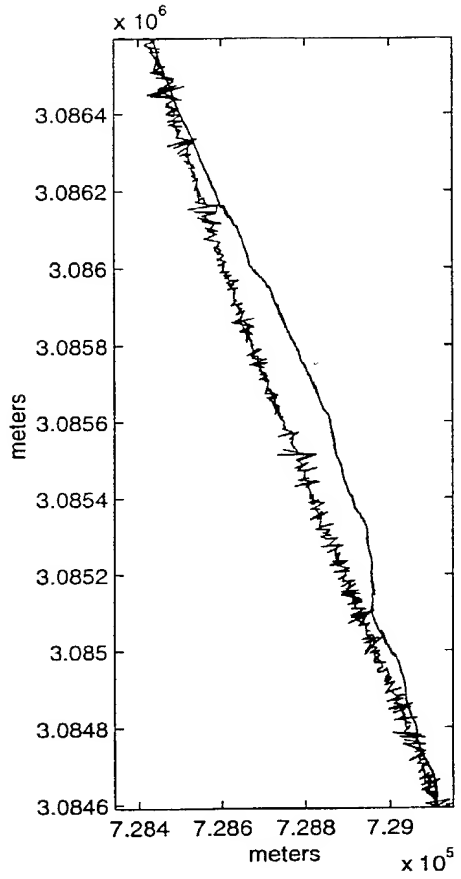

Figure 8.15: Example where lock is lost. The dotted, fuzzy line is the USBL position and the solid line the LOST2 position estimate

09905706.051101

Bibliography

[1] Sobel, D., *Longitude: The True Story of a Lone Genius Who Solved the Greatest Scientific Problem of His Time.* Penguin Books, 1995.

[2] Leonard, J. J. and Durrant-Whyte, H. F., *Directed Sonar Sensing for Mobile Robot Navigation.* Boston:Kluwer Academic Publishers, 1992.

[3] J. G. Bellington, e., *Scientific and Environmental Data Collection with Autonomous Underwater Vehicles.* MIT Sea Grant Marine Industry Collegium, 1992.

[4] Bellingham, J. G. and Consi, T. R., "Robots underwater - ongoing research at mit," *Sea Technology*, March 1990.

[5] Bergem, O., "A multibeam sonar based positioning system for an auv," *Eighth International Symposium on Unmanned Untethered Submersible Technology*, September 1993.

[6] Storkerson, N., Kristensen, J., Indreeide, A., Seim, J., and Glancy, T., "Hugin - uuv for seabed surveying," *Sea Technology*, February 1998.

[7] Brown, D. K., "Damn the mines!," *Proceedings of the U.S. Naval Institute*, March 1992.

[8] Beckman, R., Martinez, A., and Bourgeois, B., "Auv positioning using bathymetry matching," *Oceans*, 2000.

[9] Tesch, G., "Acoustic-based position discrimination of a moving robot," *Oceans*, 2000.

[10] Elfes, A., "Sonar based real-world mapping and navigation," *IEEE Journal of Robotics and Automation*, 1987.

[11] Lucido, L., Pesquet-Popescu, B., Opderbecke, J., Rigaud, V., Deriche, R., Zhang, Z., Costa, P., and Larzabal, P., "Segmentation of bathymetric profiles and terrain matching for underwater vehicle navigation," *International Journal of Systēms Science*, 1998.

[12] Feder, H., Leonard, J., and Smith, C., "Adaptive sensing for terrain aided navigation," *Oceans*, 1998.

0990706.051101

[13] Feder, H., Leonard, J., and Smith, C., "Adaptive concurrent mapping and localization using sonar," *In Proc. IEEE Internation Conference on Robots and Systems*, 1998.

[14] E. S. Maloney, e., *Dutton's Navigation and Piloting.* Annapolis, MD: Naval Institute Press, 1985.

[15] Kuristky, M. and Goldstein, M., *Autonomous Robot Vechicles.* Springer-Verlag, 1990.

[16] Joannides, M. and Gland, F. L., "Position estimation of a towed underwater body," *Proceedings of the 32nd IEEE Conference on Decision and Control*, 1993.

[17] Syck, J. M., *Dynamics of Undersea Cables.* PhD dissertation, University of Miami, 1980.

[18] Blondel, P. and Murton, B. J., *Handbook of Seafloor Sonar Imagery.* John Wiley and Sons, 1997.

[19] Larsen, M., "Synthetic long baseline navigation of underwater vehicles," *Oceans*, 2000.

[20] Stokey, R. P. and Austin, T., "The navicomputer: A portable long baseline navigation system designed for interface to an autonomous underwater vehicle," *Oceans, 2000*, 2000.

[21] Austin, T., Stokey, R., and Sharp, K., "Paradigm: A buoy-based system for auv navigation and tracking," *Oceans*, 2000.

[22] "Simrad hipap simrad hpr 400 series."

[23] E. Geyer, J. D., P. Creamer and Gains, R., "Charateristics and capabilities of navigation systems for unmanned untethered submersibles," *Proceedings of International Symposium on Unmanned Untethered Submersible Technology*, 1987.

[24] May, M. B., "Gravity navigation," *Record of the 1978 Postition, Localization and Navigation Symposium*, November 1978.

[25] Tuohy, S., Leonard, J., Bellingham, J., Oatrikalakis, N., and Chryssostomidis, C., "Map based navigation for automonomous underwater vehicles," *International Journal of Offshore and Polar Engineering*, March 1996.

[26] Tyren, C., "Magnetic anomalies as a reference for ground-speed and map-matching navigation," *The Journal of Navigation*, May 1982.

[27] Moravex, H. P. and Elfes, A., "High resolution maps from wide angle sonar," *Proceedings IEEE Int. Conf. Robotics and Automation*, 1985.

0990706.051101

[28] Thrun, S., Fox, D., and Burgard, W., "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning*, 1998.

[29] Thrun, S., Gutmann, J.-S., Fox, D., Burgard, W., and Kuipers, B., "Intergrating topological and metric map for mobile robot navigation: A statistical approach," *AAAI-98*, 1998.

[30] Olson, C. F., "Probabilistic self-localization for mobile robots," *IEEE Transactions on Robotics and Automation*, 2000.

[31] Huttenlocher, D., Klanderman, G., and Rucklidge, W., "Comparing images using the hausdorff distance," *IEEE Transaction on Pattern Analasis and Machine Intellegnce*, September 1993.

[32] Betge-Brezetz, S., Hebert, P., Chatila, R., and Devy, M., "Uncertain map making in natural enviroments," *Proc. IEEE Conf. robotivs and Automation*, April 1996.

[33] Feder, H., Leonard, J., and Smith, C., "Adaptive concurrent mapping and localization using sonar," *Proc. IEEE int. Workshop on Intelligent Robots and Systems*, October 1998.

[34] Moutarlier, P. and Chatila, R., "Stochastic multisensory data fusion for mobile robot location and environment modeling," *5th Int. Conf. on Robotics Research*, 1989.

[35] Smith, R., Self, M., and Cheeseman, P., *Autonomous Robot Vechicles.* Springer-Verlag, 1990.

[36] Bar-Shalom, Y. and Fortmann, T. E., *Tracking and Data Association.* Academic Press, 1988.

[37] Gelb, A. C., *Applied Optimal Estimation.* The MIT Press, 1973.

[38] Feder, H. J., *Simultaneous Stochastic Mapping and Localization.* PhD dissertation, Massachusetts Institute of Technology, 1999.

[39] Brooks, R., "The whole iguana," *Robotics Science*, 1989.

[40] Mataric, M. J., "Enviroment learning usisg a distributed representation," *Proceedings IEEE Int. Conf. on Robotics and Automation*, 1990.

[41] Mataric, M. J., "Intergration of representation into goal-driven behavior-based robots," *Proceedings-IEEE Int. Conf. on Robotics and Automation*, 1992.

[42] Mataric, M. J., "Bahavior-based control: Examples from navigation, learning and group behavior," *Journal of Experimental and Theoretical Artificial Intelligence*, 1997.

0990706.051101

[43] Kaelbling, L., Cassandra, A., and Kurien, J., "Acting under uncertainity: Discrete baysesian models for mobile robot navigation," *Proceedings IEEE Int. Conf. on Intelligent Robots and Systems*, 1996.

[44] Shatkay, H. and Kaelbling, L., "Heading in the right direction," *Proceedings of the 15th int. conf. on Machine Learning*, 1998.

[45] Thau, R. S., *Reliably mapping a robot's enviroment using fast vision and local, but not global meteric Data.* PhD dissertation, Massachusetts Institute of Technology, 1997.

[46] Rodriguez, J. and Aggarwal, J., "Navigation using image sequence analysis and 3d terrain matching," *Workshop on Interpretation of 3D scenes*, 1989.

[47] Brogan, W. L., *Modern Control Theory.* Prentice-Hall, 1991.

[48] Cox, D. and Hinkley, D., *Theoretical Statistics.* Chapman and Hall, 1974.

[49] Casella, G. and Berger, R. L., *Statistical Inference.* Duxbury Press, 1990.

[50] Boots, B. N. and Getis, A., *Point Pattern Analysis.* SAGE Publications, Inc., 1988.

[51] Allison, P. D., *Survival Analysis Using the SAS System, A Practical Guide.* SAS Institute, Inc., 1995.

[52] Morgan, L. H., Martinez, A., Bourgeois, B., and Myers, L., "Distribution fitting of a regular point process," *In Proceedings from the 2001 Joint Statistical Meetings*, August 2001.

[53] Bar-Shalom, Y. and Li, X., *Estimation and Tracking: Principles, Techniques, and Software.* Artech House, 1993.

[54] Papoulis, A., *Probability, Random Variables, and Stochastic Processes, Third Edition.* McGraw-Hill, 1991.

[55] Brown, R. G. and Hwang, P. Y., *Introduction to Random Signals and applied Kalman Filtering.* John Wiley and Sons, 1997.

[56] Beckman, R. R., "Location of submerged towfish, (lost)," Master's thesis, Tulane University, 2000.

[57] Jackson, L. B., *Digital Filters and Signal Processing, Third Edition.* Kluwer Academic Publishers, 1996.

[58] Bourgeois, B. and Martinez, A., "Depthimeter - precise vessel depth for bathymetry," *Marine Geodesy*, 1999.